IBM Cloud Pak AIOps 4.2.1 - PDF 2

© Copyright IBM Corp. 2025.
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Tables of Contents

Infrastructure automation 1
- User interface 2
- Provisioning 5
  - Administration 6
    - Roles and permissions 7
    - Managing connections 11
      - Configuring an Alibaba connection 13
      - Configuring an Amazon EC2 connection 13
      - Configuring an IBM Cloud connection 13
      - Configuring an IBM Cloud Kubernetes Service 14
      - Configuring a Huawei Cloud connection 14
      - Configuring a Google Cloud connection 14
      - Configuring a vSphere connection 15
      - Configuring a Nutanix connection 15
      - Configuring an OpenStack connection 15
        - OpenStack connection 15
        - PowerVC cloud on OpenStack 15
        - z/VM Cloud Manager Appliance on OpenStack 16
      - Configuring Tencent connection 16
      - Configuring a VMware NSX-T connection 16
      - Configuring VMware vRealize Automation 16
      - Configuring a NSXv connection 17
      - Configuring an Azure connection 17
      - Configuring Ansible Integration 17
      - Configuring Microsoft HyperV connection 18
    - Using custom Terraform providers 18
    - Managing Terraform Versions 19
      - Modifying a provisioned environment 21
    - Configuring Managed services 21
      - Using an external database for the module 22
      - Configuring a custom HTTPS certificate for the console 23
      - Enabling or disabling Business Process Manager 24
      - Enabling or disabling Ansible integration 24
      - Accessing Managed services by using a custom URL 25
      - Using pre-created secrets in Managed services 26
      - Configuring Managed services to use a proxy 28
      - Configuring Managed services to import out of box templates and services using your github.com token 28
      - Configuring SSH keys and known hosts in provider terraform pod 29
    - Provisioning Virtual Machines and Instances 30
    - Creating a Service in Infrastructure Management for Virtual Machine Provisioning 79
    - Service Provisioning in Infrastructure Management using an orchestration template 82
    - Assigning a Custom Analysis Profile to a Virtual Machine 85
    - Creating a tag mapping for Infrastructure management integration 88
  - Templates 88
    - Shared parameters 88
      - Managing Data Types from the user interface 90
      - Managing Data Objects from the user interface 92
      - Prebuilt Data Type list 93
    - Creating and managing templates 93
      - Defining the structure of Managed services template 93
      - Tags module 97
      - Importing a template 98
      - Editing a template 100
      - Duplicating a template 102
      - Deleting a template 102
      - Viewing a template 102
      - Changing preloaded content tokens 103
    - Creating and managing template instances 103
      - Deploying a template 103

Managed services Kubernetes resource controller for Managed services templates 104
Viewing template instance details 107
Deleting a template instance 108
Modifying a template instance 108
Managing virtual machine resources 110
Importing existing VMware vSphere virtual machine 112
Importing an existing PowerVC or OpenStack virtual machine 113
Migrating virtual machines using VMware VMotion 113
Creating and managing snapshots 115
Services 115
Creating and managing services 116
Inbuilt services in Managed services library 116
Managing categories 118
Creating a category 118
Deleting a category 118
Bookmarking Service Library URL with preselected category 119
Creating a service 119
Prerequisites for using tags in a service 122
Working with the Parameters tab 122
Creating new input and output parameters 122
Managing existing input and output parameters 124
Managing an existing activity parameter 124
Working with Composition tab 126
Activities in the Composition tab 127
Decision 128
Email notification 129
Configuring and managing SMTP server connection for Email notifications 129
REST Hook 130
Applications 131
Templates 132
Helm charts 135
Ansible 136
Credential 136
Parameters tab of Ansible Credentials activity 138
Inventory 139
Inventory and Hosts tabs of Ansible Inventory activity 140
Job Templates 140
Parameters tab of Ansible Job Template activity 142
Pricing for service plans 143
Managing service code structure 146
Supported interpolation expressions 152
Input output mapping for templates and helm charts 154
Input and output parameter mapping for resthooks 156
Parameter mapping for decision 159
Input and output parameter mapping for email notification 159
Service instance level parameters 160
Using shared parameters in service interpolations 161
Service validations 162
Viewing service details 164
Importing a service 165
Editing a service 166
Assigning access to a service 166
Duplicating a service 167
Changing a category 167
Deleting a service 168
Saving a service to different repositories 168
Service level actions 169
Service version actions 169
Adding a service version 169
Copying a service version 169
Editing a service version 170

Publishing a service version 170
Deleting a service version 171
Retiring a service version 171
Unretiring a service version 172
Creating and managing service instances 172
Creating a service instance 172
Creating bind instances for service instances 174
Managed services Kubernetes resource controller for Managed services 175
Editing a service instance 177
Upgrading or downgrading a service instance 179
Viewing service instance details 180
Terminating and deleting a service instance 181
Resuming or retrying a service instance 181
Creating and managing custom actions 182
Creating a custom action 182
Creating a custom action category 183
Viewing custom action details 184
Editing a custom action 184
Assigning custom action to a service version 184
Assigning access to a custom action 185
Deleting a custom action 185
Managed services Content 186
Offline Content Installation 187
Deploying Starterpack and Integration Content 188
Deploying a SingleVirtualMachine Starterpack template 188
Integrating with ServiceNow CMDB 188
Deploying OpenShift Container Platform 189
Deploying Kubernetes cluster 189
Working with IBM Cloud Kubernetes Service template 190
Deploying IBM Cloud Kubernetes service template 190
Deploying helm-tiller template on IBM Cloud Kubernetes Service 191
strongSwan Helm Chart 192
Scaling nodes on IBM Cloud Kubernetes service 192
Importing Kubernetes target cluster to hub cluster 193
Managing the Content Runtime 194
System requirements 194
Content Runtime Overview 195
Enabling FIPS on Pattern Manager and Software Repository containers deployed by Advanced Content Runtime 197
Provisioning and managing your Content Runtime infrastructure 197
IBM template parameters 200
Amazon template parameters 202
VMware template parameters 204
Other template parameters 206
Use existing Chef Server when deploying Content Runtime 207
IBM with Existing Chef Server template parameters 209
Amazon with Existing Chef Server template parameters 211
VMware with Existing Chef Server template parameters 212
Other with Existing Chef Server template parameters 214
Offline Content Runtime deployment 216
Knife Command Configuration 220
Content Runtime plan and apply 221
Updating the Content Runtime Docker Containers 223
Deploying a GDPR enabled Content Runtime 223
Troubleshooting and known issues 227
Managing a software repository 232
Populating the software repository 233
Populating the Installation Manager repository 235
Installation Manager Toolkit binaries 236
Using IBM Installation Manager live repositories 237
Using IBM Fix Central 241
Setting up a composite repository 242

Validating the Installation Manager repository 242
Backup Considerations for Content Runtime 243
Updating the Content Runtime SSL Certificates 243
Understanding and Deploying Middleware Content Templates 243
Deploying a Middleware template 244
Deploying a WebSphere Application Server V9 Standalone template 245
Deploying content and content runtime from Managed services that is behind a firewall 247
Terraform Template Overview 247
Template Cloud Parameters 251
Template Middleware Parameters 253
Customizing Templates 254
Template modification 254
Template variables modification 256
Chef Guidelines 257
Modifying a deployed middleware template instance 262
Terraform Provider Configuration 262
Managing IT and infrastructure 263
Infrastructure management 264
Managing your infrastructure 264
Visualizing infrastructure 265
Managing Infrastructure and Inventory 265
Managing controls 358
Policies and Profiles Guide 358
Administration 396
Monitoring, Alerts, and Reporting 396
Chargeback 436
Managing integrations 441
Managing Providers 441
ServiceNow integration 488
Change risk integration 490
Turbonomic integration 491
Business Process Manager integration 24
Ansible integration 24
Infrastructure management to {{site.data.keyword.cp4waiops_notm}} connections 492
Migrating to IBM Cloud Pak for AIOps 494
Migrating IBM Cloud Private {{site.data.keyword.infra_mgmt_manageservices}} 494
Migrating Red Hat CloudForms and {{site.data.keyword.cp4mcm_notm}} appliances 497
Backup and restore Infrastructure Automation 500
Installing the backup and restore tools 502
Backing up Infrastructure Automation 505
Restoring Infrastructure Automation 509
Upgrading IBM Cloud Pak for AIOps backup and restore artifacts 516

Technology preview 517

APIs 518
Custom resource definitions 519
IBM Cloud Pak for AIOps CRDs 519
AIModelUI-v1 520
AIOpsEdge-v1beta1 521
AIOpsUI-v1 594
AIOpsUIExtension-v1 598
AIManager-v1beta1 598
Algorithm-v1alpha1 601
Algorithm-v1beta1 601
ASM-v1beta1.md 602
BaseUI-v1 604
BundleManifest-v1alpha1 605
BundleManifest-v1beta1 606
ConnectorUI-v1 607
InsightsUI-v1 608
Installation-v1alpha1 611
Kong-v1alpha1 612
Tunnel-v1 612

Infrastructure Automation CRDs 614
IAConfig 615
IAManageService 615
IMInstall 619
Accessing APIs 622
Enabling metrics monitoring in API gateway Kong 625
Enabling logging in API gateway Kong 627
User management and Platform UI API 633
Generating API keys for authentication 633
User management (IAM) APIs 634
Preparing to run API commands 634
User management and authentication APIs 636
Account management 636
Get information about all accounts 637
Get information about an account 637
Directory management 638
Connect to an LDAP directory 638
Get information about an LDAP directory 639
Update an LDAP directory 640
List LDAP connections 641
Delete an LDAP directory 641
Delete LDAP directory by ID 642
Search for user groups in your LDAP directory 642
Search for users in your LDAP directory 643
Import user groups from your LDAP directory 643
Import users from your LDAP directory 644
User group management 644
Get all user groups 645
Delete a user group 645
User management 646
Get information about all users 646
Get information about all members of the team to which a user belongs 646
Get teams to which a user is assigned 647
Get number of teams to which a user is assigned 647
Get directories to which a user has access 648
Get highest role that is assigned to a user across teams 649
Get highest role that is assigned to a user and CRN across teams 649
Get team resources that are assigned to a user 650
Get team resources that are assigned to a user by resource type 650
Get team resources that are assigned to a user by resource type and action type 651
Get team role mappings 651
Delete a user 652
Get a user's account information 652
Single sign-on APIs 653
Enable SAML
Export metadata 654
Import metadata 656
Verify SSO configuration status 657
Disable SAML 658
Identity Provider APIs 659
Register IdP 660
SAML with SCIM dependency registration 660
SAML registration without any dependency 661
SAML with LDAP dependency registration 662
Get IdP registration 663
Get IdP registration by name 663
Get the list of IdP registration 664
Get IdP registration by query
Updating IdP 666
Deleting IdP registration 667
IdP V3 APIs 668
Registering the OIDC clients using IdP V3 668

Deleting the registration of the client 669
Getting the list of registered OIDC clients 670
Updating the OIDC clients 672
Registering the SAML clients using IdP V3 673
SAML with SCIM dependency registration 673
SAML registration without any dependency 676
SAML with LDAP dependency registration 677
Getting SAML registration by UID 678
Updating SAML registration 681
Deleting SAML registration 682
Getting the host details 683
Different schema elements for IdP V2 683
Different schema elements for IdP V3 684
Service health and version check APIs 685
Identity manager service 685
Swagger JSON 685
Version check 686
IAM policy administration service 687
Swagger JSON 687
Health check and readiness probe 687
Version check 688
IAM token service 688
Version check 688
Licensing APIs 689
License Service data per cluster 689
API authentication 689
Obtaining and updating an API token 690
Service account token 691
APIs 694
Obtaining a status page 698
License Service Reporter data for multiple clusters 700
Obtaining and updating an API token 700
APIs 701
Issue resolution API 704
Probable cause customization API 704
Change risk API 707
Metric API 711
Automation (runbook) API 714
Topology APIs 715
Accessing topology service Swagger UI 716
Enabling access to URL routes 717
Topology API reference 717
Properties 718
Edge labels 720
Edge types 721
Entity types 723
Status (and state) 726
Timestamps 728
Topology examples 728
Virtual machine example 729
OpenStack example 731
Physical device example 737
Infrastructure Management APIs 739
API Overview 739
Prerequisites 740
REST API Authorization 740
Password Updates 741
CRUD 741
HTTP Basics 742
JSON Specification 746
Query Specification 750
Bulk Querying 753
Collection Filtering 754

Pagination 755
API Reference 756
Primary Collections 756
Actions 759
Alert Management 760
Authentication Management 762
Automate Model 765
Automate Domains 768
Automation Requests 769
Automate Workspaces 772
Categories and Tags 774
Chargebacks, Rates and Assignments 778
Cloud Volumes 802
Conditions 803
Configuration Script Source Management 805
Custom Actions 807
Custom Attributes 811
Custom Button Management 814
Event Streams 817
Flavor Management 817
Generic Object Management 818
Group Management 825
Host Management 827
Instance Management 828
Metric Rollups 830
Network Routers 831
Notifications Support 831
Orchestration Support 834
Physical Server Management 836
Picture Management 838
Policy Management 839
Policy Collections 846
Provider Support 848
Provision Requests 856
Request Support 860
Reports Support 862
Role Management 869
Security Groups 871
Service Management 872
Service Queries 882
Service Custom Actions 884
Service Dialogs 889
Service Orders 892
Setting Ownership 895
Settings Management 896
Snapshot Management 898
Tags and Tagging 899
Tasks 905
Tenant Management 906
User Management 908
Virtual Machine Management 910
API Appendices 914
Quick Reference 914
Collection Metadata 915
Resource Attributes 917
Provider Types 926
Provision Request Supported Attributes 926
Advanced Provisioning Attributes 929
SOAP to REST Mapping guide 930
Deprecated Types 933
Deprecated Resource Attributes 933

API Examples 934
Foundation 934
Querying Examples 934
Paging Queries 935
Automation Requests 937
Create an Automation Request 937
Create multiple Automation Requests 938
Providers 941
Create a Provider 941
Create a Provider with multiple credentials 942
Update a Provider 942
Refresh a Provider 943
Delete a Provider 943
Provisioning Requests 944
Create a Provision Request 945
Create multiple Provision Requests 947
Service Catalogs 954
Add a Service Catalog 954
Add multiple Service Catalogs 955
Edit a Service Catalog 955
Edit Service Catalogs 956
Assign Service Templates 956
Unassign Service Templates 957
Delete Service Catalogs 958
Service Templates 958
Edit a Service Template 958
Edit Service Templates 959
Delete Service Templates 959
Services 960
Order a Service 960
Order Multiple Services 961
Edit a Service 962
Edit a Service with Patch Method 963
Edit a Service with PUT 963
Edit Services 963
Reconfigure a Service 964
Refresh Service Template Dialog Fields 965
Refresh Service Dialog Fields 966
Retire a Service 967
Retire Multiple Services 968
Trigger a Custom Action on a Service 969
Delete Services 972
VM Management 973
Start a VM 973
Stop a VM 974
Suspend a VM 974
Scan a VM 974
Set owner of a VM 975
Add Event to a VM 975
Delete a VM 976
Tag Management 977
Assign Tags 977
Assign Tags to Service Template 978
Unassign Tags 979
Unassign Tags from Service Template 980
Usage Flows 981
Compliance Policy Flow 981
Managed services APIs 990
API reference 990
Prerequisites 991
Common APIs 993
Provider and connection 993

Template and template instance 998
Terraform version 1008
Terraform file APIs 1010
Resource specific 1011
Service and service instance 1018
Custom action APIs 1085
Dynamic parameter discovery 1096
Applications 1096
Shared parameter 1102
SMTP connection 1108
Usage specific 1111
Example: Retrieve deployed templates list 1111
Sample Python script 1113
Content APIs 1114
Upload cookbooks and roles 1115
List cookbooks and roles 1117
Clear cookbooks and roles 1118
Get service logs 1119

Reference 1120
Operator cluster service versions (CSVs) 1120
Operator permissions (IBM Cloud Pak for AIOps) 1121
Operator permissions (Infrastructure Automation) 1159
Install commands requiring root or sudo access 1168
Alert and Event schemas 1168
Alerts and Events schema fields 1171
Resource schemas 1177
EventType schemas 1178
Normalized Alert Mapping Rules 1178
Infrastructure Automation 1187
Reference - Infrastructure management 1187
Installation parameters for the Managed services 1187
Support Matrix 1191
Supported Browsers 1191
Supported Platforms 1191
Supported Providers 1192
Provisioning Support on Infrastructure Providers 1192
Feature Support on Infrastructure Providers 1193
Feature Support on Cloud Providers 1196
Feature Support on Container Providers 1201
Feature Support on Network Providers 1201
Feature Support on Configuration Management Providers 1202
Remote Console Support 1202
Operating System Support 1205
Smart State Analysis Support 1206
Supported Guest Operating Systems 1207
Deployment Planning Guide 1208
Integration with OpenShift Container Platform 1231
High Availability Guide 1238
Appliance Hardening Guide 1253
Methods Available for Automation 1264
Managing Authentication for Infrastructure Management (stand-alone) 1295
Active Directory 1295
2-Factor-Authentication with IPA 1297
IPA/AD Trust authentication 1299
LDAP authentication 1301
SAML authentication 1304
OpenID-Connect authentication 1308
Scripting Actions in Infrastructure Management (stand-alone) 1312
General Configuration 1337
Self Service User Interface Guide 1431
Reference - Managed services 1440
Password migration of Managed services 1440

Managing the usage of virtual machines 1441
Managed services pods 1441
SSH Keys and related input required while deploying content runtime 1442
Control the version of a Terraform provider 1442
Terraform CAMC provider 1442
Shipped Terraform providers 1447
Supported Terraform templates 1452
Troubleshooting 1453
Install and upgrade 1453
Clusters and nodes 1461
Connections 1468
Secure Tunnel 1478
AI model management 1480
Alert management 1486
ChatOps 1490
Topology 1493
Viewing the topology logs 1496
User management (IAM) 1497
Login 1497
Infrastructure Automation 1497
Troubleshooting installing and uninstalling Infrastructure Automation 1497
Troubleshooting Infrastructure Management 1499
After an upgrade, users cannot access Infrastructure management 1499
Error using multiple tabs in the same browser 1500
Non-LDAP users cannot access Infrastructure management 1500
The vmware-infra-* pods in CrashLoopBackOff error states 1501
VM operator encounters an error when creating CRD 1501
VM operator encounters an error when upgrading 1501
Error deleting cloud volumes 1502
Error logging out of and logging in to Infrastructure management 1502
New data is not shown from Infrastructure management 1503
Troubleshooting Managed services 1504
General issues 1505
Installation 1505
Installation of Managed services on {{site.data.keyword.roks}} 1507
CLI tool 1508
Deployment issues 1508
Security related issues 1512
Open-source support in Managed services 1512
Importing custom CA Certificate into Terraform provider pod 1513
User interface 1514
Service Composer 1515
Service library 1519
Middleware Content 1519
Content related issues and useful links 1520
Starter Library templates are not loaded 1520
SSH handshake failure 1520
Terraform provider does not support resource 1521
SSH key not found 1521
Node already registered with the Chef server 1522
Failure cleanup does not occur 1523
Known issues with Managed services and VMware vSphere 1523
Recovering the Content Runtime 1524
Bootstrap fails with knife sudo password error 1528
IBM Cloud Pak for AIOps or {{site.data.keyword.ocp}} template deployments 1528
Terraform related issues 1528
Collecting data for Managed services 1529
Refreshing internal certificates 1530
Collecting logs using camCollector script 1530
Updating the resource limits for Managed services containers 1531
Support 1532
Opening a support case 1532

Installing the MustGather tool 1533
Running the MustGather tool 1534
Command syntax 1534
Primary use cases 1534
Installation related problems 1535
Application related problems
IBM Cloud Pak for AIOps related problems 1535
Additional use cases 1535
Run script to complete corrective actions 1536
Run baseline data collection 1536
Pod specific data collection 1536
Product specific data collection 1537
Installation verification 1537
Copying pod files and directories 1537
Complex command 1537
Miscellaneous use cases 1537
Scale down the MustGather data collection 1537
Gather foundational services data
Command help 1537
Data collection modes 1537
Primary modes 1538
Secondary modes 1538
View mode 1540
Advanced tips 1540
Use manual collection mode to run commands 1540
Use CPFILES to copy files 1541
Run custom scripts 1542
Security bulletins and fixes 1546

Notices 1548

Glossary 1550

Infrastructure Automation

Infrastructure Automation is a stand-alone capability/module included within IBM Cloud Pak for AIOps. When you have a license for IBM Cloud Pak for AIOps, you are entitled to install and use Infrastructure Automation.

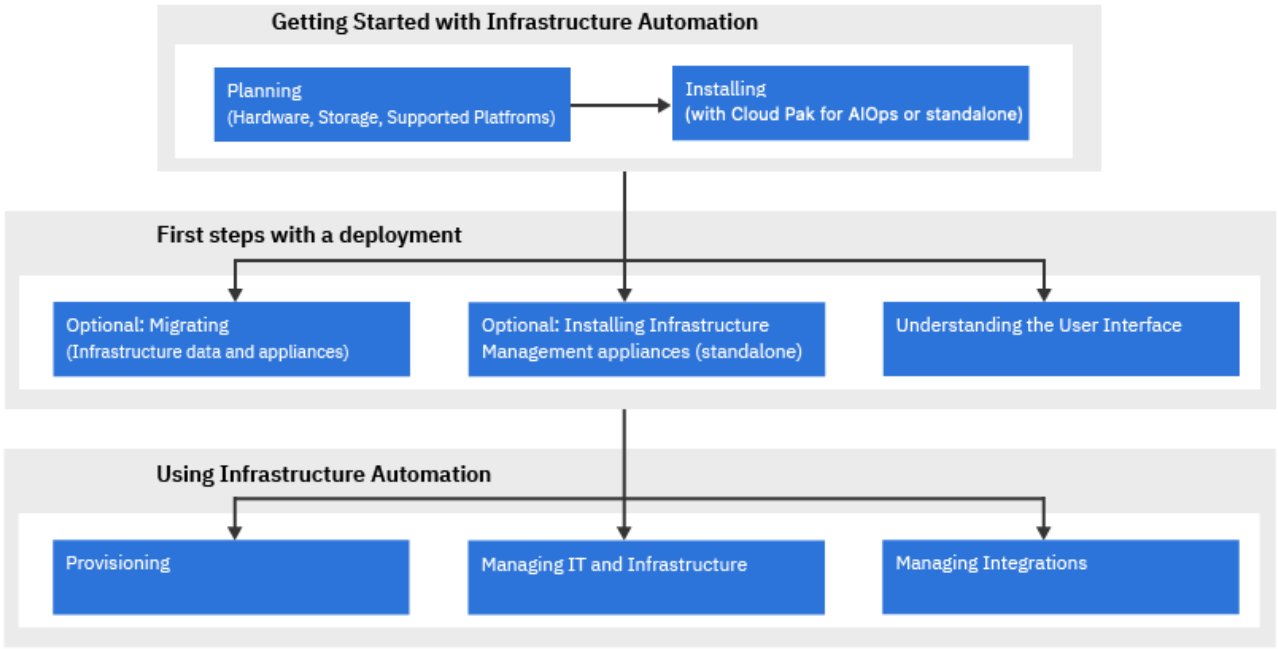
Infrastructure Automation brings together Terraform and ManageIQ for an integrated, Infrastructure Automation capability for IT Operators.

Infrastructure Automation delivers two distinct features:

- Managed services, previously called Terraform and Service Automation. This feature uses self-service capabilities to orchestrate resources.
- Infrastructure management, previously called IBM Red Hat CloudForms. This feature provides capabilities for discovering, managing, and automating the deployment of VMs, cloud services and Kubernetes clusters.

To learn more about Infrastructure Automation, click a subject within the following image to view the related available documentation (or review the subjects that are listed following the image):

Infrastructure Automation documentation flow



Getting started

Planning

Before you install IBM Cloud Pak for AIOps - Infrastructure Automation, plan your installation, configuration, and overall cluster requirements. For more information, see Planning.

Installing

Infrastructure Automation is included with IBM Cloud Pak for AIOps and can be installed for use with IBM Cloud Pak for AIOps or as a stand-alone component. For information about, see Installing Infrastructure automation.

First steps with a deployment

Migrating data from another deployment

You can migrate infrastructure data, such as services data and appliances, to Infrastructure Automation from other products and services, such as IBM Cloud Pak for Multicloud Management, IBM Cloud Private, and Red Hat CloudForms. For more information, see Migrating to IBM Cloud Pak for AIOps.

Installing Infrastructure Management appliances

With Infrastructure Automation installed, you can install Infrastructure management appliances for your environment. These appliances can be installed on different supported platforms, such as Red Hat OpenStack Platform and Red Hat Virtualization, VMware vSphere, and Amazon Elastic Compute Cloud (Amazon EC2). For more information, see Installing and upgrading appliances.

Understanding the user interface

Learn how to use the IBM Cloud Pak Automation console to access and use Infrastructure Automation tools and features. For more information, see User interface.

Infrastructure Automation use cases

Once set up, you and your DevOps teams can use Infrastructure Automation to provision, integrate, and manage IT and infrastructure.

Provisioning

Infrastructure Automation provide the capability to automate provisioning of infrastructure and virtual machine applications using Terraform and Ansible. Provisioning IT brings infrastructure-as-code support and helps with sharing templates across multiple instances and with provisioning repeatable infrastructure with confidence. This infrastructure-as-code support helps to provide you with a standardized environment for your developers. For more information, see Provisioning.

Managing It and infrastructure

Infrastructure Automation includes supported versions of Terraform that works across multiple cloud environments like RHEV, VMWare, OpenStack and multiple cloud providers, like AWS, Microsoft Azure, Google, and IBM Cloud. For more information, see Managing IT and infrastructure.

Managing integrations

With Infrastructure Automation, you can enable default integrations with ServiceNow, Ansible Tower, Turbonomic, and Business Process Manager. Once integrations are setup, the cloud infrastructures can be provisioned through the Infrastructure Automation catalog or through APIs. Changes and operations to the provisioned infrastructures go through Infrastructure Automation, which can synchronize the changes with other systems. For more information, see Managing integrations.

Infrastructure Automation UI console views

Use the Infrastructure Automation UI console to view, create, and automate the management of your infrastructure.

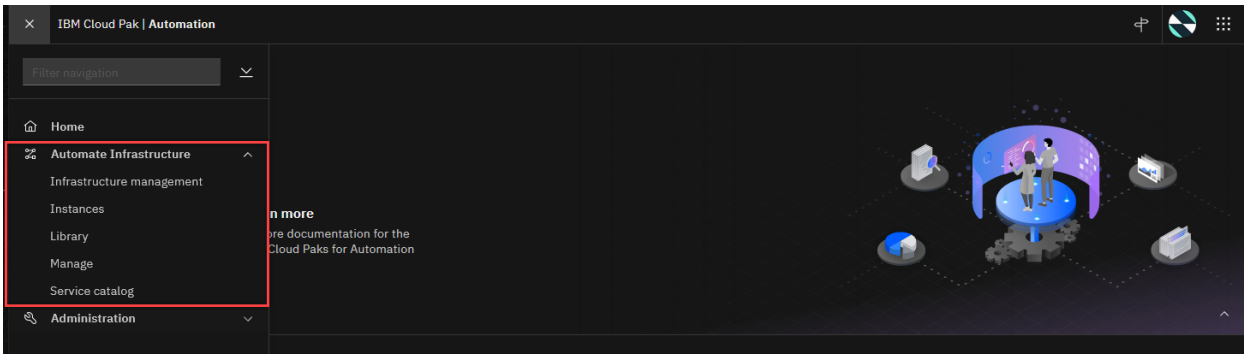


Figure. Main page

Infrastructure management

Instances

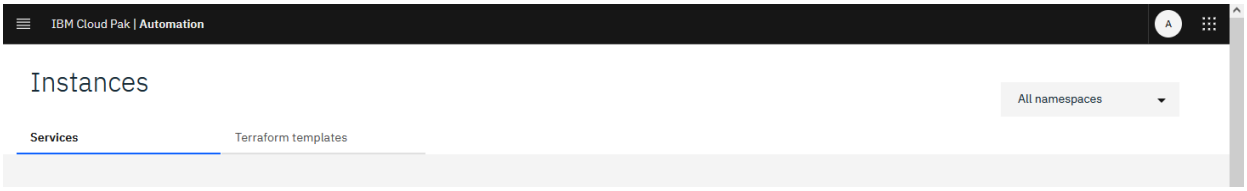


Figure. Instances

Services

On the *Services* tab, you view service instances. If you need to create a service instance, you are redirected to the **Library** > **Services** page, where you can create the service instance.

For more information, see Services.

Terraform templates

On the *Services* tab, you view Terraform instances. If you need to create a Terraform instance, you are redirected to the **Library** > **Terraform templates** page, where you can create the Terraform instance.

For more information, see Templates.

Library

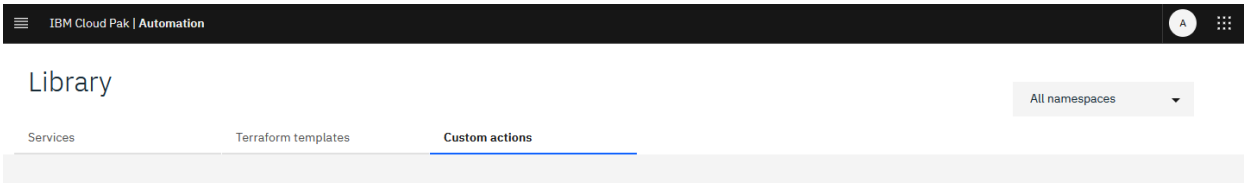


Figure. Library page

Services

On the *Services* tab, you can view all services, which can be sorted by category. You can also choose to create or import services.

For more information, see Services.

Terraform templates

On the *Terraform templates* tab, you can view all templates. You can also choose to import a template.

For more information, see Templates.

Custom actions

On the *Custom actions* tab, you can view all custom actions, which can be sorted by category. You can also choose to create different custom actions.

For more information, see Creating and managing custom actions.

Manage

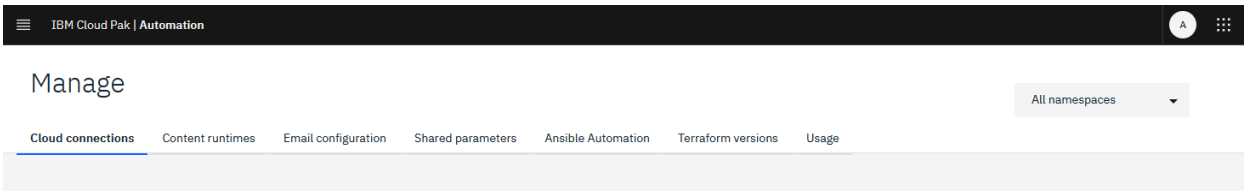


Figure. Manage page

Cloud connections

On the *Cloud connections* tab, you can view and create cloud connections. Cloud connections allow you to create connections to each of your target cloud providers that are used when deploying instances.

For more information, see Managing connections

Content runtimes

On the *Content runtimes* tab, you can view and create advanced content runtimes.

For more information about creating and using advanced content runtimes, see Managing content runtime

Email configuration

On the *Email configuration* tab, you can view and create email server configurations. To use email-related features within Manage services you need to configure an email server to send these emails.

For more information about configuring and managing SMTP server connection for email notifications, see Managing Email Configuration

Shared parameters

On the *Shared parameters* tab, you can view and create data types and data objects that can be used as shared parameters. Shared parameters define shared configuration to store data and are used by templates. Shared parameters allow you to create reusable data objects. This helps save time when you are deploying new instances and creating services.

For more information, see Shared parameters.

Ansible Automation

On the *Ansible Automation* tab, you can create and view Ansible connections. By creating a globally accessible connection to Ansible Tower, you can include Ansible Automation within your services.

For more information about configuring an Ansible Tower connection, see Configuring the Ansible Tower connection.

Terraform versions

On the *Terraform versions* tab, you can view versions of Terraform that can be offered when deploying templates and services. You can also create versions, if needed. A default version can be set, which is used when the template metadata does not specify a version.

For more information, see Managing Terraform Versions.

Usage

On the *Usage* tab, you can view the highest number of virtual machines (VMs) managed by Manage services each month. You can view the current and monthly usage, and download the usage data.

For more information, see Managing the usage of virtual machines.

Service catalog

The Service catalog view displays the deployable services and applications and related details. It also displays the deployed instances and related details. With the service catalog you can complete the following tasks:

- As an IT administrator, you can define a new service by using the service composer, test it and when it is ready, publish it to the Service catalog. You can decide which parameters to expose to users, such as to hide the complexity around provisioning from users.
- As a user, such as a developer, you can access the Service catalog to provision infrastructure and applications.

The Service catalog view can be refreshed at particular interval by using the **`Autorefresh`** menu. The default is 10 seconds. You can also disable the autorefresh.

To access the Service catalog, go to IBM Cloud Pak for AIOps console and click **Automate infrastructure** > **Service library**.

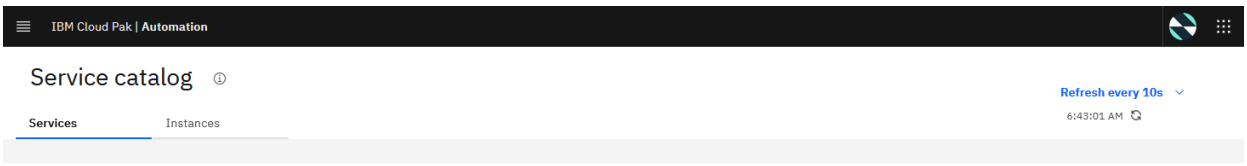


Figure. Service catalog page

- Services
- Instances

Services

On the *Services* tab, a tiled list of published services and applications is displayed. The list of services and applications is sorted according to categories. Clicking the service deploys that service to your environment. You can search a service by using **Search services** option or search by category.

For more information about services, see the following topics:

- Creating a service
- Publishing a service
- Deploying a service

Instances

On the *Instances* tab, a list of all deployed service and application instances is displayed. If needed, you can use the search box to filter the listed instances.

For each instance, the following information is displayed:

- The name of the instance
- The name of the service
- The status of the instance, Executed or Terminated
- The date and time when the service was deployed

A typical operation user can access all the following instance actions from the overflow menu:

- **View**: For more information, see Viewing service instance details.
- **Edit**: For more information, see Editing a service instance.
- **Custom actions**: For more information, see Custom actions.
- **Retry/resume** : For more information, see Resuming or retrying a service instance.
    - Applicable only for `Error` state instances.
    - `Retry` tries to deploy the instance from the beginning of the execution.
    - `Resume` tries to resume deploying the instance from the point of failure.
- **Upgrade/downgrade**: For more information, see Upgrading or downgrading a service instance
- **Terminate**/**Delete**: For more information, see Terminating and deleting a service instance.
    - Only `Active` instances can be terminated.
    - Only `Terminated` instances can be deleted.

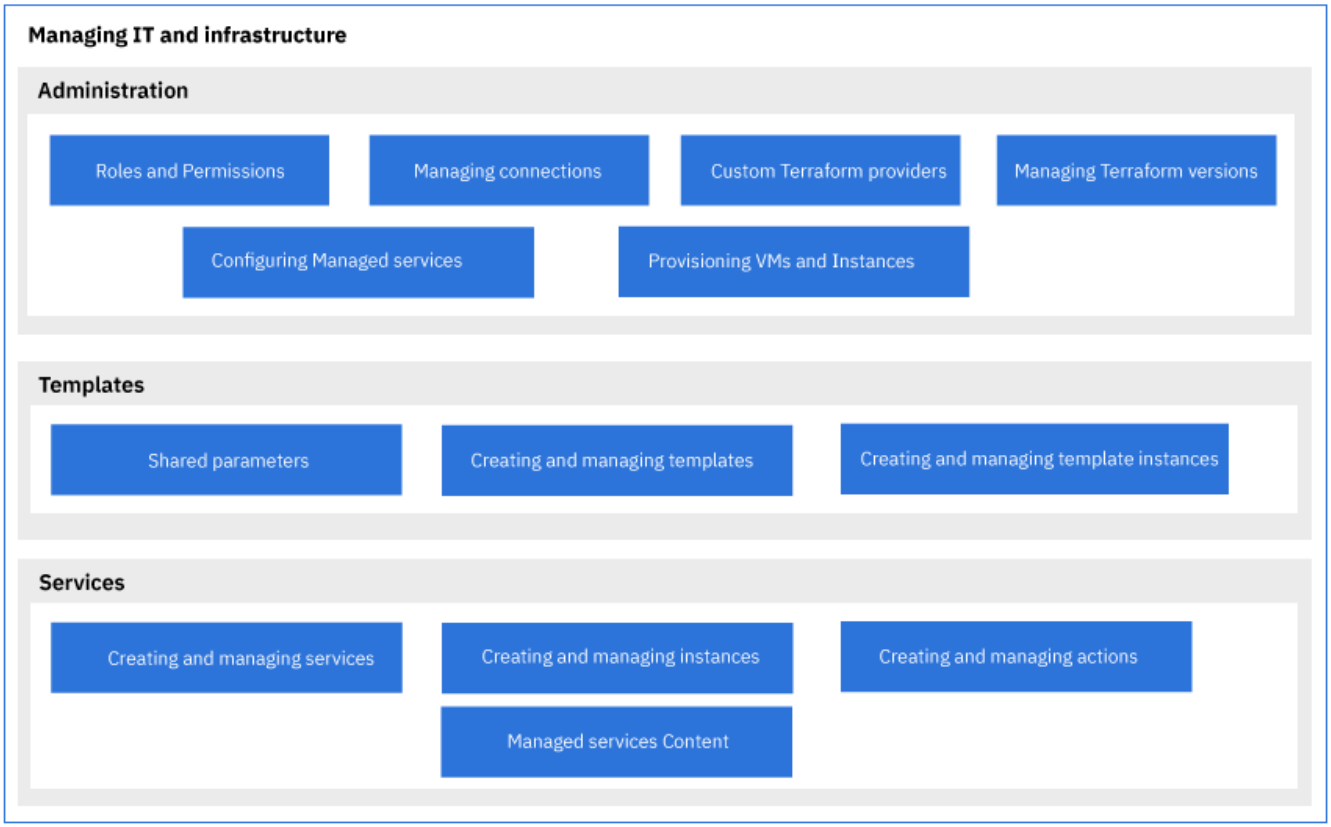
Provisioning

Provisioning IT brings infrastructure-as-code support and helps with sharing templates across multiple instances and with provisioning repeatable infrastructure with confidence.

The Manage Services compose complex full stack services with ease using Terraform and Ansible.

The provisioning IT also provides RBAC controlled self service access for developers to provision the infrastructure they need.

To learn more about provisioning IT for Infrastructure Automation, click a subject within the following image to view the related available documentation (or click the subject from list that follows the image):



- Adminstration
  - Roles and permissions
  - Managing connections
  - Custom Terraform providers
  - Managing Terraform Versions
  - Configuring Managed services
  - Provisioning Virtual Machines and Instances
- Templates
  - Shared parameters
  - Creating and managing templates
  - Creating and managing template instances
- Services
  - Creating and managing services
  - Creating and managing service instances
  - Creating and managing custom actions
  - Managed services Content

Provisioning administration

- Roles and permissions for Infrastructure Automation
- Managing connections
- Custom Terraform providers
- Managing Terraform versions
- Configuring Managed services
- Provisioning Virtual Machines and Instances
- Creating a Service in Infrastructure Management for Virtual Machine Provisioning
- Service Provisioning in Infrastructure Management using an orchestration template
- Assigning a Custom Analysis Profile to a Virtual Machine
- Creating a tag mapping for Infrastructure management integration

Roles and permissions for Infrastructure Automation

For accessing and completing actions with IBM Cloud Pak® for AIOps Infrastructure Automation, users require specific roles and permissions. Review the different available roles, permissions, and the associated actions that users can complete with the roles and permissions. These roles and permissions help you get your users set up to begin day to day operations.

- Predefined roles
- Creating a role
- Permissions

Predefined roles

A role defines the permissions that a user or group has. You can edit the default roles or create new roles if the default set of permissions in a role does not align with your business needs. For more information, see Managing roles.

The following roles are available by default:

- Administrator
- Automation Administrator
- Automation Analyst
- Automation Developer (AIOps Developer role)
- Automation Operator
- User

Notes:

- The User predefined role is listed as available by default but this role does not include any permissions for completing actions within IBM Cloud Pak for AIOps. This default role is used within the IBM® Automation family of offerings, which includes IBM Cloud Pak for AIOps, however, this roles is not used within IBM Cloud Pak for AIOps.

  As this role does not include or provide any permissions within IBM Cloud Pak for AIOps, do not assign the role to users within IBM Cloud Pak for AIOps. If more IBM Cloud Paks are installed, this roles can have permissions for completing actions within tools for those other IBM Cloud Paks.

- Not all permissions are associated with a predefined role. A user with with a permission to manage roles, such as the Administer platform permission, can assign permissions to a role when needed.

Default user

The default user (`admin`) is automatically assigned the following roles when the roles are added to the platform:

- Administrator

Creating a role

When adding a user, you can choose to create a role instead of assigning an existing role. You can also choose to directly create a role. The options for creating a role are the same regardless of the path you select to create the role.

To directly create a role, complete the following steps:

1. Log in to the console as an administrator with permissions to manage users.
2. From the main navigation, click **Administration** > **Access control**.
3. Select the **Roles** tab and click **New role**.
4. Enter the details for the new role:
   - Name - Enter a distinctive and descriptive name for the role.
   - Description - Enter an optional description for the role to define the role, such as to identify the objective or responsibilities for users with the role.
5. Click **Next**.
6. Select the permissions for the role. You can use the *Find permissions* search field to find and filter the list of permissions.
7. Click **Next**.
8. Review the details for the role and click **Create**.

Permissions

The following table describes the actions that are associated with each permission.

Infrastructure Automation

Table: User administration permissions

Permissions	Actions	Automation Developer	Automation Operator	Automation Analyst	Automation Administrator	Administrator
View Kubernetes resources	(1)			X		
Edit Kubernetes resources	(2)	X				
Manage Kubernetes resources	(3)		X			
Administer Kubernetes resources	(4)				X	
Administer Kubernetes namespace	(5)					X

1. View Kubernetes resources in the Cloud Pak namespaces.
    - View resources
2. Edit Kubernetes resources in the Cloud Pak namespaces.
    - View resources
    - Edit resources
3. Manage Kubernetes resources in the Cloud Pak namespaces.
    - View resources
    - Edit resources
    - Create resources
4. Administer Kubernetes resources in the Cloud Pak namespaces.
    - View resources
    - Edit resources
    - Create resources
    - Delete resources
5. Administer Kubernetes namespace
    - Administer Kubernetes namespace (administrator ClusterRole)
    - Access Administration Hub
    - Manage identity providers
    - Manage teams
    - Manage service IDs

Platform administration

Permissions in this category enable an administrator to configure, customize, monitor, and manage the platform.

Only the Administer platform permission is associated with a predefined role (Adminstrator). This permission includes all other platform administration permissions.

Table: Platform administration permissions

Permissions	Actions	Automation Developer	Automation Operator	Automation Administrator	Administrator
Administer platform	(1)				X
Manage configurations	(2)				
Manage platform health	(3)				
View platform health	(4)				

1. Administer platform

    This permission offers the most comprehensive set of actions for managing and monitoring the platform. Users with this permission have elevated privileges and can grant or revoke all permissions, including other administrative permissions. Users with this permission can complete the same actions as users that have the following permissions:

    - Manage configurations (Administer platform)
    - Manage platform health (Administer platform)
    - Manage platform roles (User administration)

- Manage users (User administration)
- Manage user groups (User administration)
- Manage service instances (Service instances)

2. Manage configurations

   Users with this permission can customize the platform, integrate the platform with other applications, and enable connections to unsupported data sources. Users with this permission can access the Customizations page, the Configurations page, and the JDBC drivers tab on the Platform connections page. Some actions require specific services to be installed. Users with this permission can complete the following actions:

   - Configure connection to SMTP server

     Note: Requires Watson Studio or Watson Knowledge Catalog. An SMTP connection enables the platform to send emails.

   - Configure integration with IBM Guardium appliances

     Note: Requires Watson Knowledge Catalog. Use IBM Guardium to audit access to sensitive data on remote databases.

   - Configure connections to Hadoop clusters

     Note: Requires Execution Engine for Apache Hadoop.

   - Customize branding
   - Enable and disable home page cards
   - Enable and disable default support links
   - Add and delete custom support links
   - Enable and disable guided tours
   - Import JDBC drivers

     Note: Requires the common core services. JDBC drivers enable users to connect to unsupported data sources.

3. Manage platform health

   Users with this permission can monitor resource use, set quotas and alerts, manage workloads to maintain the health of the platform, and gather diagnostic data when problems occur. Users with this permission can access the Monitoring page and the Diagnostics page. Users with this permission can complete the following actions:

   - Monitor workloads and resource use

     Note: Workloads include active runtimes and jobs. Resources include memory and vCPU use across services, instances, and environments.

   - Stop any runtime environment

     Note: Environments are hardware and software configurations defined for running analytical assets or jobs.

   - View pod status, details, and logs

     Note: Services are composed of Kubernetes pods. A pod is an instance of a process that runs on the cluster.

   - Restart pods
   - View platform quotas and service quotas

     Note: A quota specifies the maximum amount of memory or vCPU that the platform or a service should use.

   - View event history and alerts
   - Set and edit platform resource quotas
   - Set and edit individual service resource quotas
   - Create and run diagnostics jobs

     Note: A diagnostic job gathers information that can be used for troubleshooting problems.

   - Delete diagnostics jobs

4. View platform health

Users with this permission can monitor resource use and workloads across the platform to gauge the health of the platform. Users with this permission have read-only access to the Monitoring page. Users with this permission can complete the following actions:

- Monitor workloads and resource use

  Note: Workloads include active runtimes and jobs. Resources include memory and vCPU use across services, instances, and environments.

- View pod status, details, and logs

  Note: Services are composed of Kubernetes pods. A pod is an instance of a process that runs on the cluster.

- View platform quotas and service quotas

  Note: A quota specifies the maximum amount of memory or vCPU that the platform or a service should use.

- View event history and alerts

Service instances

A service instance is a specific deployment of a service. Some services can be deployed more than once. Some service instances have their own access controls.

Table: Service instances permissions

Permissions	Actions	Automation Developer	Automation Operator	Automation Administrator	Administrator
Create service instances	(1)				X
Manage service instances	(2)				

1. Create service instances

   Users with this permission can create service instances and storage volumes. The types of service instances depend on the services that are installed. Users with this permission can complete the following actions:

   - Create service instances

     Note: If an instance has access controls, the user who creates the instance is an instance administrator.

2. Manage service instances

   Users with this permission can manage access to any service instance or delete any service instance from the Instances page. Users with this permission can complete the following actions:

   - Create service instances
   - View all service instances
   - Add users to any service instance
   - Assign an instance role to instance users
   - Remove users from a service instance
   - Delete any service instance

What permissions do I have?

You can see what permissions you have from your profile. Your permissions are determined by the roles that are assigned to you.

To see what roles are assigned to you:

1. Log in to the IBM Cloud Pak Automation console and click your avatar in the upper right of the toolbar.
2. Click **Profile and settings**.
3. Click the **Roles** tab.

The permissions that are associated with your role (or roles) are listed in the **Enabled permissions** column.

Adding users

To add users, you need to have an external LDAP server set up for creating user profiles and authentication. When a user record is included within the LDAP server, you can add the user account within the IBM Cloud Pak Automation console. You can add a user directly or as part of an LDAP group. For more information, see Adding users.

Assigning roles and permissions

When you add a user or group, you must specify the role and permissions that they have. These roles and permissions can be granted to users through a combination of the IBM Cloud Pak Automation console and an external LDAP server for managing user authentication.

Red Hat OpenShift Container Platform roles

To add and manage users and roles for accessing and using the Red Hat OpenShift Container Platform web console, you need to use the Red Hat role-based access control (RBAC). For more information, see the Red Hat OpenShift topic Using RBAC to define and apply permissions.

Managing connections

In the Connections view, manage your connections to define the Connections you want to set up.

Managing connections:

- Add a connection
- Search for a connection
- Edit a connection
- Testing a connection
- Delete a connection

Add a connection

To add a connection, complete the following steps:

1. In the navigation bar, click **Manage** > **Cloud connections**.
2. Click **Create connection**.
3. Select the cloud provider to which you want to connect. The following cloud providers are supported:
   - Alibaba Cloud
   - Amazon EC2
   - IBM
   - IBM Cloud Kubernetes Service
   - Google Cloud
   - Huawei Cloud
   - Microsoft Azure
   - Microsoft HyperV
   - Nutanix
   - OpenStack
   - Tencent Cloud
   - VMware NSX-T
   - VMware NSXv
   - VMware vRealize Automation
   - VMware vSphere
   - Other

   Note: Apart from the values available in the provider type drop-down list, a generic provider type that is named `Other` can be used. You do not have to explicitly define a cloud connection of type `Other` because it is provided implicitly by the system. It does not define any connectivity attributes and must be used for cases where the cloud connection details are captured inside the provider section of the template itself.
4. In **Select a namespace** > **Assign access** section, select either of the following values based on your requirement:
   - **Make connection globally accessible** - Connections created globally are accessible by all users.
   - **Make connection part of a namespace** - Associate the connection with a specific namespace. For this option, you must select a **Namespace** from the drop-down list to which the connection must be assigned. The namespace that is present on the **Cloud connections** main page is preselected and you can modify if needed.

   Note: If you create a cloud connection in a specific namespace, then only deployments to that namespace can use the cloud connection.
5. Enter the connection name and description, and specify the required connection parameters depending on the cloud provider you selected.

   Note: The connection name must begin with a letter and must have a combination of only these characters: [a-zA-Z][a-zA-Z0-9_-]

   Managed services must have access to the provider cloud region where you want to deploy your templates. The access is granted by supplying Managed services with access credentials. The method for obtaining access credentials depends on the cloud provider.

For information about specifying an Alibaba Cloud Configuring an Alibaba Cloud connection

For information about specifying Amazon EC2 access credentials, see Configuring an Amazon EC2 connection.

For information about specifying IBM Cloud access credentials, see Configuring an IBM Cloud connection.

For information about specifying IBM Cloud Kubernetes Service credentials, see Configuring IBM Cloud Kubernetes Service.

For information about specifying Huawei Cloud credentials, see Configuring a Huawei Cloud connection.

For information about specifying Azure access credentials, see Configuring an Azure connection.

For information about specifying Nutanix credentials, see Configuring a Nutanix connection.

For information about specifying OpenStack access credentials, see Configuring an OpenStack connection

For information about specifying Tencent Cloud Configuring Tencent Cloud connection

For information about specifying VMware vRealize Automation credentials, see Configuring VMware vRealize Automation.

For information about specifying vSphere access credentials, see Configuring a vSphere connection.

For information about specifying VMware NSX-T access credentials, see Configuring a VMware NSX-T connection.

For information about specifying NSXv access credentials, see Configuring an NSXv connection.

For information about specifying Microsoft HyperV credentials, see Configuring Microsoft HyperV connection

Note: If you require a more secure environment for connections then see Create an EgressFirewall.

6. Click **Create**. The connection is added to the connection list.

Search for a connection

In **Cloud connections** page, you can use the following methods to search for connections:

- You can select a namespace from the **All namespaces** drop-down list. The list of cloud connections in the page changes based on the selected namespace. You can also select "All namespaces" to see all cloud connection records for which you have access.
- You can enter text in the search text to filter the connection list.
- You can use the show filter icon next to search text to filter records based on type or namespace. The namespace option has **Show globally accessible** option, which refers to those connections that are globally accessible across all namespaces and are not assigned to any specific namespace.
- You can click the **Name** header to sort the connection records.
- You can even combine these search mechanisms for better results.
- The pagination option allows you to limit the number of records displayed at a time.

Edit a connection

To edit an existing connection, follow these steps:

1. In the navigation bar, click **Manage** > **Cloud connections**.
2. In the connection list, click the connection that you want to edit.
3. Modify the existing connection details and parameters.

   Important: You can change a namespace to globally accessible but cannot assign a different namespace for a connection. You can also change a globally accessible namespace to a different namespace but be informed that with this namespace change you might not be able to perform any operations on existing deployments.

4. Click **Update**.

Testing a connection

To test a connection, follow these steps:

1. In the navigation bar, click **Manage** > **Cloud connections**.
2. In the connection list, click the menu in the **ACTIONS** column for the connection that you want to test.
3. In the action list, click **Test connection**. If the connection is successful, a green check mark is displayed in the **ACTIONS** column for the connection.

Delete a connection

To delete a connection, follow these steps:

1. In the navigation bar, click **Manage** > **Cloud connections**.
2. In the connection list, click the menu in the **ACTIONS** column for the connection that you want to delete.
3. In the action list, click **Delete connection**. A confirmation window is opened.
4. Click **Delete** to confirm that you want to delete the connection.

Note: Remember to remove any EgressFirewall rules you may have added for this connection. See Create an EgressFirewall.

Configuring an Alibaba cloud connection

In order for Managed services to access your Alibaba Cloud environment, you must provide the value of the following parameters:

- Access Key ID - This is the Alicloud access key. This is a required field.
- Access Key Secret - This is the Alicloud secret key. This is a required field.
- STS Session Token - This is the Alicloud Security Token Service. This is an optional field.
- Region - This is the Alicloud region. This is a required field.

Configuring an Amazon EC2 connection

In order for Managed services to access your Amazon Web Services (AWS) account, an access key and the secret access key to your account are required.

The access key and the secret access key are not your AWS user name and password, but they are special tokens that allow Managed services to communicate with your AWS account through secure API calls. You can also have temporary access key, access secret, and session token that expires in a specified amount of time. Refresh the credentials in this cloud connections page after the temporary credentials expire. Alternatively, you can define the access key, access secret, and session token in the **`Provider`** section:

```
provider "aws" {
 region = "${var.aws_region}"
 token  = "${var.aws_token}"
 access_key = "${var.aws_access_key}"
 secret_key = "${var.aws_secret_key}"
}
```

For more information about creating AWS access keys, see Creating, Modifying, and Viewing Access Keys (AWS Management Console).

For information about the available resources, see AWS Provider.

Configuring an IBM Cloud connection

For Managed services to access your IBM Cloud environment, specify the IBM Cloud API key to deploy IBM Cloud services and specify IBM Cloud User and IBM Cloud Classic Infrastructure API Key to deploy IBM Cloud classic infrastructure like compute instances.

- **IBM Cloud API Key** - Enter the IBM Cloud API key that is associated with your user identity.
- **IBM Cloud Classic Infrastructure User Name** and **IBM Cloud Classic Infrastructure API Key** fields - Enter the IBM Cloud user name and the IBM Cloud classic infrastructure API key values respectively.

To retrieve the **IBM Cloud Classic Infrastructure User Name**:

1. Log in to IBM Cloud Portal.
2. Click **Manage** > **Access (IAM)** > **Users**.
3. Select the user from the list. The **User details** tab of **Manage user** page is displayed.
4. The **User name** value, a combination of a **`number_emailID`** is listed under **VPN Password** section.

To retrieve the **IBM Cloud Classic Infrastructure API Key** values:

1. Log in to IBM Cloud Portal.
2. Click **Manage** > **Access (IAM)** > **API keys** > **In View** dropdown, select **Classic infrastructure API keys** >.

3. In the returned list, select additional action icon (vertical ellipses) > **Details** > **Copy the API key**.

To create the **IBM Cloud API Key** values:

1. Log in to IBM Cloud Portal.
2. Click **Manage** > **Access (IAM)** > **API keys** > **In View** dropdown, select **My IBM Cloud API Keys** >.
3. Click **Create an IBM Cloud API key** to create a new one.

For more details, see IBM Cloud IAM.

For information about the available Terraform resources, see IBM Cloud Provider.

Note: If you created templates with an old version that uses `terraform-provider-ibm` older than v0.4, be aware that the resource names for the IBM Cloud provider were changed. For more information, see Migrating to v0.4.

Configuring an IBM Cloud Kubernetes Service

For Managed services to access kubernetes clusters, you must provide the value of the following parameters:

Note: You must have administrator privileges for the deployment of IBM Cloud Kubernetes Service cluster and charts into that cluster. If you do not have the privileges, contact your administrator.

- **IBM Cloud API Key** - Enter the IBM Cloud API Key. It is the key to access your Kubernetes cluster.
- **IBM Cloud region name** - Enter the IBM Cloud region name where your cluster is running.
- **IBM Cloud Kubernetes Service cluster name** - Enter the Kubernetes cluster name.
- **Kubernetes Helm version** - Select the Kubernetes Helm version on which the cluster is running.

The cluster credentials are validated before creating a cloud connection. However, the health of the cluster is not validated.

If you want to know the details of your cluster, do the following steps:

1. Go to bluemix console at https://console.bluemix.net/dashboard/apps/.
2. Click **Containers**.
3. From the **Containers** page menu, click **Clusters**.
4. In the Clusters page, select the **Location** from the drop-down menu or enter your region name in the **Filter** to search. You can find a list of all clusters including the clusters that were deployed with Managed services.
5. Click your cluster to view its details.

For information about IBM Cloud Kubernetes Services, see IBM Cloud Kubernetes Service.

Configuring a Huawei Cloud connection

For Managed services to access your Huawei Cloud server, specify the authentication URL, user name, and password of the Huawei Cloud server.

- **Authentication URL** - The authentication url of Huawei Cloud server.
- **User Name** - The user name to connect to Huawei Cloud server.
- **Password** - The password of the Huawei Cloud user.
- **Domain Name** - The project domain name.
- **Region** - The region where the resources are created.
- **Project Name** - The tenant or project name.

For more information about Huawei Cloud, see Huawei documentation.

Note: The Huawei cloud connection may fail even with correct credentials. However, this error does not prevent you from using the connection for template deployment.

Configuring Google Cloud connection

For Managed services to access your Google Cloud account, upload account credentials, enter ID of the Google Cloud project and the region to deploy the service.

For **Service Account Credentials**, upload a JSON file that is used to describe account credentials. It is downloaded from Google Cloud console.

For more information about Google Cloud, see Google Cloud documentation.

Configuring a vSphere connection

In order for Managed services to access your vSphere environment, you must provide the IP address or host name of the vCenter server and the user credentials to access it. Managed services supports vCenter version 6.0 or later.

For more information about vSphere, see the VMware vSphere 6.0 Documentation Center.

What to do next

Deploy a vSphere template. For more information, see Importing a template and Deploying a template.

For information about the available resources, see VMware vSphere Provider and Resource Documentation.

Configuring a Nutanix connection

For Managed services to access your Nutanix server, specify the user name and password of the Nutanix server.

- **Nutanix User** - The user to connect to Nutanix.
- **Nutanix Password** - The password of the Nutanix user.
- **Nutanix Server** - The Nutanix server IP or host name.
- **Nutanix Port** - The Nutanix port. The default value is 9440.
- **Allow insecure connections** - Select true or false value. This indicates whether insecure connections are allowed in the Nutanix server.

For more information about Nutanix AHV (on-prem) offering, see Nutanix AHV Admin Guide.

Configuring an OpenStack connection

Managed services supports OpenStack cloud connection, PowerVC on Openstack connection, and z/VM Cloud Manager Appliance on OpenStack connection.

- OpenStack connection
- PowerVC cloud on OpenStack
- z/VM Cloud Manager Appliance on OpenStack

OpenStack connection

Managed services supports OpenStack cloud connection. For Managed services to access your OpenStack environment, you must provide the value of the following OpenStack cloud connection parameters:

- **Authentication URL** - Enter the authentication URL. It is used for the purpose of identification. The syntax is `http://IP_ADDRESS/identity/v3`. Here, v3 refers to version 3 of the keystone identity API.
- **User Name** - Enter the user name that is used to log into OpenStack server.
- **Password** - Enter the password of the OpenStack user.
- **Domain Name** - Enter the keystone domain name.
- **Region** - Enter the region of the cloud OpenStack.
- **Project Name** - Enter the name of the project that is supported by OpenStack and is within the specified domain.

Note: The values for the URL, domain, and region can be discovered on any OpenStack implementation by examining the output of 'OpenStack catalog list'.

For more information about these values, see OpenStack Provider.

PowerVC cloud on OpenStack

Managed services can manage PowerVC cloud V1.3.3 or higher on OpenStack. For more information about PowerVC, see PowerVC documentation.

For Managed services to access your PowerVC OpenStack environment, you must provide the value of the following connection configuration parameters:

- **Authentication URL** - Enter the authentication URL. It is used for the purpose of identification. The syntax is `https://<hostname_or_ip-address>:5000/v3`. Here, v3 refers to version 3 of the keystone identity API.
- **User Name** - Enter the user name that is used to log into OpenStack server.
- **Password** - Enter the password of the OpenStack user.
- **Domain Name** - Enter the keystone domain name. Default is the PowerVC cloud default value.
- **Region** - Enter the region of the cloud OpenStack.
- **Project Name** - Enter the name of the project that is supported by OpenStack and is within the specified domain.

For more information about these values, see OpenStack Provider.

z/VM Cloud Manager Appliance on OpenStack

Managed services can now manage IBM z/VM 7.1 virtual machines using the IBM Cloud Infrastructure Center.

About the task

For more information about IBM Cloud Infrastructure Center, see Documentation. You must configure OpenStack cloud connection.

For Managed services to access your IBM Cloud Infrastructure Center OpenStack environment, you must provide the value of the following parameters:

- **Authentication URL** - Enter the authentication URL. It is used for the purpose of identification. The syntax is http://IP_ADDRESS:5000/v3. Here, v3 refers to version 3 of the keystone identity API.
- **User Name** - Enter the user name that is used to log into OpenStack server.
- **Password** - Enter the password of the OpenStack user.
- **Domain Name** - Enter the keystone domain name. Default is the IBM Cloud Infrastructure Center default.
- **Region** - Enter the region of the cloud OpenStack. RegionOne is the IBM Cloud Infrastructure Center default.
- **Project Name** - Enter the name of the project that is supported by OpenStack and is within the specified domain.

For more information about these values, see OpenStack Provider.

Configuring Tencent cloud connection

In order for Managed services to access your Tencent Cloud environment, you must provide the value of the following parameters:

- Secret ID - This is the TencentCloud secret id.
- Secret Key - This is the TencentCloud secret key.
- Region - This is the TencentCloud region.

Configuring a VMware NSX-T connection

For Managed services to access your VMware NSX-T server, specify the user name, password, and manager host details.

- **NSX_T User** - The user name to connect to VMware NSX-T server.
- **NSX_T Password** - The password of the VMware NSX-T user.
- **NSX_T Manager Host** - The VMware NSX_T manager host name or IP address.
- **Allow insecure connections** - Indicates whether insecure connections are allowed in VMware NSX-T server. You must set this value to True.

For more information about VMware NSX-T, see VMware NSX-T documentation.

Configuring VMware vRealize Automation

For Managed services to access your VMware vRealize Automation environment, you must provide the details of the server.

- **vRA User** - The user name to log in to the vRA server.

- **vRA Password** - The password for the user name to log in to the vRA server.
- **vRA tenant** - The vRA portal tenant.
- **vRA Host** - The path (IP or fully qualified host name) to the end point of vRA REST APIs (usually the vRA server). For example, https://myvra.example.com/.
- **Insecure** - Enter "true" to make an insecure connection to the vRA server. The default value is "False".

For more information about VMware vRealize Automation, see VMware Terraform provider for vRealize Automation.

Configuring an NSXv connection

For Managed services to access your NSX environment, you must provide the IP address of the NSX manager and the user credentials to access it.

What to do next

Deploy an NSXv template. For more information, see Importing a template and Deploying a template.

For information about the available resources, see NSXv Terraform Provider and Resource Documentation.

Configuring an Azure connection

In order for Managed services to access your Azure environment, you must provide the value of the following parameters:

- `subscription_id`
- `client_id`
- `client_secret`
- `tenant_id`

For information about getting these values, see Creating Credentials in the Azure Portal.

What to do next

Deploy an Azure template. For more information, see Importing a template and Deploying a template.

For information about the available resources, see Microsoft Azure Provider.

Configuring the Ansible connection

Before you begin

- Make sure the Ansible Integration is activated from the Managed services. See Enabling or disabling Ansible Integration.
- Make sure the variable **Prompt On Launch** is checked for the Credential and Inventory on the Ansible for the job templates that are created in the Ansible and used in the Managed services integration.

Procedure

From the Managed services to access your Ansible set up, specify username, password, and URL of the Ansible.

1. In the navigation bar, click **Manage** > **Ansible automation**.
2. Click **Create connection**.
3. Enter the following values:
   - **Ansible connection name** - Enter a unique name for the Ansible connection.
   - **Ansible user** - Enter the username of the Ansible.

     Note: The Ansible user needs to have access to the Ansible artifacts that are consumed in the designed Managed services service.
   - **Ansible password** - Enter the password for the Ansible username.

- **Ansible URL** - Enter the URL of the Ansible.

4. Click **Create** to create and test the connection.
5. Click **Test** from the overflow menu to test the Ansible connection.

Note: The user can create one Ansible connection per namespace and only one global connection that is available to all namespaces.

Configuring Microsoft HyperV cloud connection

In order for Managed services to access the Microsoft HyperV Cloud environment, you must provide value for the following parameters:

- Microsoft HyperV username - The user to connect to Microsoft HyperV. It's a mandatory field.
- Microsoft HyperV password - The password to connect to Microsoft HyperV. It's a mandatory field.
- Microsoft HyperV Host - Microsoft HyperV hostname or IP address. It's a mandatory field.
- Port - The port on which winrm runs on Microsoft HyperV system (port that is used by the Microsoft HyperV APIs to communicate with the system). The default value is 5985. It's a mandatory field.
- Https enabled - Boolean value field that indicates whether communication to winrm on the Microsoft HyperV system (used by the Microsoft HyperV API) is over http or https. The default value is https.
- TLS disabled - Boolean value field that indicates whether TLS Verification is used for the Microsoft HyperV API calls. The default value is TLS verification disabled.
- NTLM enabled - Boolean value field that indicates whether winrm on Microsoft HyperV system (used by the Microsoft HyperV API) uses NTLM authentication. The default value is NTLM authentication enabled.
- Connection timeout - The timeout to wait for the connection to become available for Microsoft HyperV API calls. The default value is 30s. It's a mandatory field.
- CA certificate path - The path to the CA certificate to use for Microsoft HyperV API calls. It's an optional field.
- Certificate path - The path to the certificate to use for authentication for Microsoft HyperV API calls. It's an optional field.
- Certificate private key path - The path to the certificate private key to use for authentication for Microsoft HyperV API calls. It's an optional field.
- TLS server name - The TLS server name for the host used for Microsoft HyperV API calls.

Uploading certificates

You must upload the certificates to the provider terraform pod by using the Create a Terraform file API with **`type=file`**. The API uploads the certificates to /home/terraform/.terraform.d/cam. The value for certificate path variables must be /home/terraform/.terraform.d/cam/your_certificate_name.

Microsoft HyperV provider plugin binary

Microsoft HyperV provider is classified by Hashicorp as community provider. The provider binary is not included with Managed services since it is neither official nor verified provider, .

If your Managed services installation has access to the internet, then the provider will be downloaded when you run the Microsoft HyperV Terrafrom template.

If your Managed services installation is in an air tight environment, then you need to download the provider binary from Microsoft HyperV Terrafrom Github repository.

If you installed Managed services on IBM Power, then you need to download the source from Microsoft HyperV Terrafrom GitHub repository and build the binary for IBM Power.

Custom Terraform providers

The Shipped Terraform providers page lists the Terraform provider plug-ins that is shipped with the product. If the provider that you need is missing, then you can directly Bring Your Own (BYO) providers into the Terraform pod volume. The file naming convention for the binaries is as follows:

- For provider plug-in: **`terraform-provider-<NAME>_vX.Y.Z`**
- For provisioner plug-in: **`terraform-provisioner-<NAME>_vX.Y.Z`**

Terraform relies on file names to determine plug-in types, names, and versions.

Follow the below steps to copy the third-party Terraform provider to the cam-provider-terraform pods:

- Create an archive with plug-in or plug-ins. The archive is expanded in the `/home/terraform/.terraform.d/plugins/` folder on provider terraform pods when the pods are restarted.

Example of a plug-in archive that contains the ibm-cloud terraform plug-in:

```
unzip -l ibm_cloud_plugin.zip
Archive: ibm_cloud_plugin.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2021-10-12 13:34   registry.terraform.io/
        0  2021-10-12 13:34   registry.terraform.io/ibm-cloud/
        0  2021-10-12 13:34   registry.terraform.io/ibm-cloud/ibm/
        0  2021-10-12 13:34   registry.terraform.io/ibm-cloud/ibm/1.33.1/
        0  2021-10-12 13:36   registry.terraform.io/ibm-cloud/ibm/1.33.1/linux_amd64/
 60833792  2021-10-01 09:08   registry.terraform.io/ibm-cloud/ibm/1.33.1/linux_amd64/terraform-provider-
ibm_v1.33.1
     5816  2021-10-01 09:13   registry.terraform.io/ibm-cloud/ibm/1.33.1/linux_amd64/README.md
    88983  2021-10-01 09:13   registry.terraform.io/ibm-cloud/ibm/1.33.1/linux_amd64/CHANGELOG.md
    15977  2021-10-01 09:13   registry.terraform.io/ibm-cloud/ibm/1.33.1/linux_amd64/LICENSE
---------                     -------
 60944568                     9 files
```

When you upload this archive file by using the API and restart the pod, the **`terraform-provider-ibm_v1.33.1`** binary is laid out under **`/home/terraform/.terraform.d/plugins/registry.terraform.io/ibm-cloud/ibm/1.33.1/linux_amd64`**.

- Use the Terraform file API to upload the plug-in archive.
- Restart **`cam-provider-terraform-api`** pods. If you have enabled the isolation mode for Provider Terraform runtime during the installation, then you must restart the **`cam-provider-terraform-runtime`** pods.

The location of the plug-in provider binary varies based on the Terraform version a provider plug-in supports:

Terraform version	Location of plugin provider binary
Terraform engine 0.12.x or lower	`/home/terraform/.terraform.d/plugins/`
Terraform engine 0.13.0 or higher and is listed in the Terraform official registry	`/home/terraform/.terraform.d/plugins/registry.terraform.io/<namespace>/<type>/<version>/linux_<arch>`
Terraform engine 0.13.0 or higher and is provided by IBM Cloud Pak for AIOps	`/home/terraform/.terraform.d/plugins/registry.ibm.com/<namespace>/<type>/<version>/linux_<arch>`
Terraform engine 0.13.0 or higher and is built inhouse	`/home/terraform/.terraform.d/plugins/<your.registry.com>/<namespace>/<type>/<version>/linux_<arch>`

Managing Terraform Versions

The Managed services component supports multiple Terraform engine versions that are offered when deploying the templates. The following Terraform engine version is shipped with the platform:

- 1.1.3

Terraform engine versions are shipped with the platform, but you can add your own versions. Terraform default version can be changed using these instructions.

The list displays the Version, the type of version and the status. The versions that are shipped with the Managed services are listed as **`System`** and the user created versions are listed as **`User`**.

Only an Administrator or Cluster Administrator can create/delete/edit Terraform versions.

The Terraform versions are not namespace specific.

Managing Terraform Versions:

- Create a version
- Search for a version
- Rename a version
- Disable a version
- Enable a version
- Set a default version
- Delete a version

Create a version

Before creating a Terraform version in Managed services, follow these steps to get the Terraform version:

1. Download and extract the Terraform binary of the version you want.
2. Rename the Terraform binary to include the version. For example, for Terraform version 0.12.16 rename `terraform` to `terraform_0.12.16`.
3. Archive the file.

To create a new version, follow these steps:

1. In the navigation bar, click **Manage** > **Terraform versions**.
2. Click **Create version**.
3. Enter the version name in the **Terraform version name** text box.
4. Click **Upload** and select your terraform archive.
5. Check **Set as default** if this version should be considered in case template metadata does not specify a version.
6. Click **Create**. The new version is added to the version list.

You must restart your `cam-provider-terraform-api` and `cam-provider-terraform-runtime` pods after you finish the above steps.

Search for a version

In **Terraform versions** page, you can use the following methods to search for versions:

- You can enter text in the search text to filter the version list.
- You can click the **Version** header to sort the versions.
- The pagination options allows you to limit the number of records displayed at a time.

Rename a version

To rename an existing version, follow these steps:

1. In the navigation bar, click **Manage** > **Terraform versions**.
2. From the overflow menu of a version, click **Rename**.
3. Modify the existing version name.
4. Click **Rename**.

Disable a version

To disable a version, follow these steps:

1. In the navigation bar, click **Manage** > **Terraform versions**.
2. From the overflow menu of a version, click **Disable**. The default version can not be disabled.
3. Click **Disable**.

   Note: This disabled version cannot be used in the future template deployments.

Enable a version

To enable a version, follow these steps:

1. In the navigation bar, click **Manage** > **Terraform versions**.
2. From the overflow menu of a version, click **Enable**.
3. Click **Enable**.

Set a default version

To set a default version, follow these steps:

1. In the navigation bar, click **Manage** > **Terraform versions**.
2. From the overflow menu of a version, click **Set as default**.
3. Click **Set as default** to confirm that you want to set this version as default.

   Note: This default version is used as a primary selection while deploying the template.

Delete a version

Only the user defined Terraform versions can be deleted. Also if the version is set as default, the delete option is disabled. If you still want to delete this version, select other version as a default version.

To delete a version, follow these steps:

1. In the navigation bar, click **Manage** > **Terraform versions**.
2. From the overflow menu of a version, click **Delete**. A confirmation window is opened.
3. Type **Delete** to confirm that you want to delete the version.
4. Click **Delete**.

   Note: The deleted version cannot be used in the future template deployments.

Modifying a provisioned environment from your local Template Designer

From the locally installed Template Designer user interface, you can modify a template and apply those changes to an existing environment.

Procedure

1. In a locally installed Template Designer, go to **Repositories** and select your repository and branch.
2. Go to **Blueprints** and open your project to provision.
3. Click **Provision**. The **Provision Environment** page is displayed. To modify an existing environment, select **Modify**. To deploy a new environment, select **Deploy**.

   Note: When **modify** is selected, the location is set to the location given when the environment was originally provisioned. You cannot change it during a **Modify** task.
4. Select the **Orchestration Service** from the drop-down list.
5. Select the environment to modify from the **Existing Environment Name** drop-down list.
6. Click **Next Edit Variables**.
7. In the **Edit Variables** tab, click **Variables** to expand the section.
8. Update any variable values that you want to change.
9. Here, you can click **Next Plan Changes** to see the plan results or you can skip the step and directly **Apply Changes**.
10. If you click **Next Plan Changes**, you can see the **Plan Output** with the status as success or failed. You can use the **Copy Log** to copy the plan summary to the clipboard.
11. If you are satisfied with the plan summary, click **Next Apply Changes** to apply the changes. A message appears after a successful plan and apply of the template.
12. Click the message to go to the **Environment** list page.
13. If you do not see your environments in the list, select the appropriate orchestration service from the **Environment** drop-down list.
14. Click the environment name to view the environment details. You can also click the **Log File** tab to view the log file for the most recent activity. Initially, the environment status is in progress and then, after successful deployment, it changes to running.

Post-installation configuration of the Managed services

Optional configuration steps after Managed services installation.

- Using an external database for the module
- Configuring a custom HTTPS certificate for the console
- Enabling or disabling Business Process Manager
- Enabling or disabling Ansible integration
- Accessing Managed services by using a custom URL

- Using pre-created secrets in Managed services
- Configuring Managed services to use a proxy
- Configuring Managed services to import out of box templates and services using your github.com token
- Configuring SSH keys and known hosts in provider terraform pod

Using an external database for Managed services

You can configure Managed services to use an external MongoDB. This configuration is optional.

External MongoDB connection

The supported version for external MongoDB is 4.2.x.

Steps to connect an external MongoDB connection:

1. Edit **IAConfig**.

   ```
   oc edit IAConfig -A
   ```

2. Update **IAConfig** using the following spec for Managed services.

   ```
   spec:
     imagePullSecret: ibm-entitlement-key
     infraAutoComposableComponents:
     - enabled: true
       name: ibm-management-im-install
     - enabled: true
       name: ibm-management-cam-install
       spec:
         manageservice:
           global:
             externalDb: <true | false >
             externalDbConfig: < external DB config in base 64 >
             externalMongoDBCertificate: <CA certificate  in  base 64 >
             mongoDbUrl: <mongodburl in  base 64  >
     license:
       accept: true
     storageClass: rook-cephfs
     storageClassLargeBlock: rook-ceph-block
   ```

   - externalDb: Set this value to true.
   - externalDbConfig: Add base64 encode for extra configuration. This is added as an object and contain all the options that require to connect the mongo connection.

   Sample configuration:

   ```
   {
     "ssl":true,
     "sslValidate":false,
   }
   ```

   Note: Add **certType: 'crt'** for crt type certification or **certType: 'pem'** for pem type certification in the external database configuration.

   - externalMongoDBCertificate: base64 encode of the certificate.
   - mongoDbUrl: External database url.

   Note: Some of the mongo instance may use only url and need not require any certificate. In such case, update externalDb and mongoDbUrl.

3. Save your **IAConfig** changes.
4. All the pods are deleted and newly created pods points to the external MongoDB connection.

Disconnection of external MongoDB and reconnect to the cam-mongo pods

Steps to disconnect from an external MongoDB connection and reconnect to the cam-mongo pods:

1. Edit the **IAConfig**.

   ```
   oc edit IAConfig -A
   ```

2. Set the externalDb to false in **IAConfig**.

```
global:
     externalDb:  false
```

3. Save your **IAConfig** changes.

4. All the pods are deleted and newly created pods points to the cam-mongo pods.

When you are unable to access Managed services routes with correct externalDB configuration, then increase the **icpdata_addon_version** to a higher value in the configmap **cam-proxy-zen-extension**.

```
oc -n cp4aiops edit configmap cam-proxy-zen-extension
```

Configuring a custom HTTPS certificate for Managed services Management Console

Procedure to install a custom certificate for use by the Managed services management console. You must first install Managed services and then customize your secrets using instructions in Procedure to configure your custom HTTPS certificate postinstallation.

You can configure your custom HTTPS certificate either during installation of Managed services or update the certificate postinstallation. Before you follow the below procedure, you need to create a secret to store your custom certificates required by the nginx server. The secret must contain the following key values.

- A key named **tls.crt** whose value must be the tls certificate.
- A key named **tls.key** whose value must be the tls key.

After creating the secret, follow one of the following procedures:

- Procedure to configure your custom HTTPS certificate during installation
- Procedure to configure your custom HTTPS certificate postinstallation

Procedure to follow to configure your custom HTTPS certificate during installation

To set the custom HTTPS certificate during installation, use the name of a new secret which contains the custom certificates as the value for **camProxy.customNginxSSLCertificate.customNginxSSLCertName** in the **IAConfig** resource specification. For more details on how to create **IAConfig** resource, see Online install of Infrastructure Automation for IBM Cloud Pak for AIOps or Online installation of Infrastructure Automation.

The following sample yaml shows **IAConfig** resource specification with customized certificate name as value for **camProxy.customNginxSSLCertificate.customNginxSSLCertName**.

```
cat << EOF | oc apply -f -
kind: IAConfig
apiVersion: aiops.ibm.com/v1alpha1
metadata:
  name: ibm-ia-installer
  namespace: cp4aiops
spec:
  imagePullSecret: ibm-entitlement-key
  infraAutoComposableComponents:
    - enabled: <Set true to install Infrastructure Management component of
{{site.data.keyword.infra_automation}} false otherwise>
      name: ibm-management-im-install
      spec: {}
    - enabled: <Set true to install Managed services component of {{site.data.keyword.infra_automation}}
false otherwise>
      name: ibm-management-cam-install
      spec:
        manageservice:
          camProxy:
            customNginxSSLCertificate:
              customNginxSSLCertName: mycert
  license:
    accept: <Set true to accept the license>
  storageClass: <Storage Class name that supports RWX>
  storageClassLargeBlock: <Select a storage class with a large block size (for example, 64k)>
EOF
```

To verify whether cam-proxy uses your custom secret, the **type** value for volume **cam-nginx-cert** must be set to your secret name once the cam-proxy pod comes up. If your secret name is **mycert**, then the type column must display **mycert**.

Procedure to configure your custom HTTPS certificate postinstallation

To set the custom HTTPS certificate after installation, do the following:

1. In Red Hat OpenShift Container Platform console, navigate to your installed ManagedService custom resource YAML.
2. Find string **customNginxSSLCertName** in the YAML.
3. Set the secret name of your custom certificate (tls.crt and tls.key) as the value for **customNginxSSLCertName**. If your secret name is **mycert** then your modified YAML looks as below:

```
camProxy:
  replicaCount: 1
  resources:
    limits:
      cpu: 500m
      memory: 1500Mi
    requests:
      cpu: 100m
      memory: 256Mi
  customNginxSSLCertificate:
    customNginxSSLCertName: mycert
```

4. Save your changes.
5. Your cam-proxy pod will be restarted. To verify whether cam-proxy uses your custom secret, once the cam-proxy pod comes up, the **type** value for volume **cam-nginx-cert** must be set to your secret name. If your secret name is **mycert**, then the type column must display **mycert**.

Enabling or disabling IBM Business Process Manager

Post the installation of Managed services, you can enable or disable Business Process Manager from the IBM Cloud Pak for AIOps.

Procedure

Complete these steps from your OpenShift console.

1. Click **Operators** > **Installed Operators**.
2. Select **cp4aiops** namespace from the **Project** drop-down list.
3. Select the operator **IBM Management Infrastructure managed services**.
4. Select the tab Managed services.
5. Edit the Managed services instance by selecting **Edit ManageService** from the overflow menu.
6. Search **camBpmProvider** in YAML and change the following values:
   - Change **replicaCount** to 1.
   - Edit **bpmEndpoint** with the BPM endpoint value.
   - Edit **bpmUserName** with the user name value.
   - Edit **bpmUserPwd** with the password.

     These are the sample values for BPM provider:

```
camBpmProvider:
  bpmEndpoint: 'https://9.9.9.9:9443/'
  bpmUserName: 'admin'
  bpmUserPwd: 'passw0rd'
  replicaCount: 1
```

7. Save the YAML file.

Note: The supported version of IBM Business Process Manager is 8.5.7.

Enabling or disabling Ansible Integration

Post the installation of Managed services, you can enable or disable Ansible integration from the IBM Cloud Pak for AIOps.

Procedure

Complete these steps from your OpenShift console.

1. Click **Operators** > **Installed Operators**.
2. Select `cp4aiops` namespace from the **Project** drop-down list.
3. Select the operator `IBM Management Infrastructure managed services`.
4. Select the tab Managed services`.
5. Edit the Managed services instance by selecting **Edit ManageService** from the overflow menu.
6. Search `camAnsibleProvider` in YAML and change the `replicaCount` to 1.
7. Save the YAML file.

Accessing Managed services by using a custom URL

You can access Managed services by using a custom URL.

Before you begin

An ingress must be defined in your cluster.

About the task

You need to perform the procedure from a node that can run OpenShift Client (oc) commands.

Procedure

1. Set up variables with necessary information.

   ```
   FIP=<Hostname>
   ```

   Where `<Hostname>` is the IBM Cloud Pak for AIOps hostname or IP.

   ```
   OAUTH2_CLIENT_REGISTRATION_SECRET=$(oc -n ibm-common-services get secret platform-oidc-credentials -o
   yaml | grep OAUTH2_CLIENT_REGISTRATION_SECRET | awk '{ print $2}' | base64 --decode)
   ```

   ```
   TSA_WLP_CLIENT_ID=$(oc -n cp4aiops get secrets cam-oauth-client-secret -o yaml -o yaml | grep CLIENT_ID
   | awk '{ print $2}' | base64 --decode)
   ```

2. Verify whether the variables have values.

   ```
   echo $FIP
   ```

   ```
   echo $OAUTH2_CLIENT_REGISTRATION_SECRET
   ```

   ```
   echo $TSA_WLP_CLIENT_ID
   ```

3. Download Managed services OIDC registration using the following command:

   ```
   curl -kvv -X GET -u oauthadmin:$OAUTH2_CLIENT_REGISTRATION_SECRET
   https://$FIP/idauth/oidc/endpoint/OP/registration/$TSA_WLP_CLIENT_ID | python -m json.tool > tsa-oidc-
   registration.json
   ```

4. Modify the resulting `tsa-oidc-registration.json` file.

   1. Remove `client_id_issued_at` and `registration_client_uri` attributes.
   2. Under the `redirect_uris` section, add the following entries:

      ```
      "https://<custom_ingresss_or_hostname_and_port>/auth/liberty/callback"
      ```

      ```
      "https://<custom_ingresss_or_hostname_and_port>/cam/auth/liberty/callback"
      ```

      Note: To access the Managed services by using a domain name and Managed services port, replace `<custom_ingresss_or_hostname_and_port>` with `<domain_name>:<port>`. If you created an ingress to access Managed services, replace `<custom_ingresss_or_hostname_and_port>` with the ingress hostname and port.

5. Update Managed services OIDC registration.

```
curl -kvv -X PUT -u oauthadmin:$OAUTH2_CLIENT_REGISTRATION_SECRET -H "Content-Type: application/json" -d
@tsa-oidc-registration.json https://$FIP/idauth/oidc/endpoint/OP/registration/$TSA_WLP_CLIENT_ID
```

Using pre-created secrets in Managed services

During the installation of Managed services, a secret named **cam-secure-values-secret** is automatically created. This secret contains the default database credentials, encryption credentials, and GitHub credentials to import default templates (IBM Cloud Pak for AIOps 2.3.x fix pack 2 onwards) and proxy information. You can override the default secret values at the time of IBM Cloud Pak for AIOps installation or post-installation by using the following procedure.

Managed services installation specification has an option **secureValues.secretName**. By default, the secret name is empty as IBM Cloud Pak for AIOps automatically creates a secret named **cam-secure-values-secret** at the time of installation. The values in this secret are used by the Managed services deployments. To override the default secret values, you must first create a new secret and then provide the name of your secret in the **secureValues.secretName** section during installation.

If you want to change the values in default secret **cam-secure-values-secret** post-installation, then create a new secret, edit the ManagedService custom resource that was created during installation and provide the name of the new secret as value for **secureValues.secretName** parameter.

You must first install Managed services and then customize your secrets using instructions in Change the secret post installation.

Procedure to change the secret value during installation and post-installation

To change the secret value either during installation or post-installtion, you first need to create a new Kubernetes secret resource with the set of secret values required by the Managed services.

Create a new Kubernetes secret resource

Complete the following steps to create a secret by using a YAML file. Your secret must contain all the data that is shown in the sample yaml file.

1. Log in to your Red Hat® OpenShift® Container Platform cluster by using the **oc login** command:

   ```
   oc login -u kubeadmin -p <YOUR_PASSWORD> --server=<YOUR_SERVER_API_URL>
   ```

2. Encode the secret values in base64. Depending on the data that you are providing, you might need to encode the HTTP and HTTPS proxy URLs, the MongoDB URL and password, and the encryption password.

   IBM Cloud Pak for AIOps 2.3.x fix pack 2 onwards, a new secret data **contentGitHubTokens** is provided to import the default templates and services by using your organizations github.com token. For more information, see Configuring Managed services to import out of box templates and services by using your github.com token

   Use the following command to encode the values to base64.

   ```
   echo "<value-to-be-encoded>" | tr -d '\n' | base64
   ```

   Here is an example command to encode the HTTP proxy URL:

   ```
   echo "http://user1:pass1@some.proxy.com:1234" | tr -d '\n' | base64
   ```

   Note: The `tr -d '\n'` removes newline character before base64 encoding. Example output:

   ```
   aHR0cDovL3VzZXIxOnBhc3MxQHNvbWUucHJveHkuY29tOjEyMzQ=
   ```

3. Define the **my-cam-secure-values-secret.yaml** YAML file by using any editor.

   ```
   vi my-cam-secure-values-secret.yaml
   ```

   Following is a sample content of the YAML file. Replace the values with your base64 encoded values:

   ```
   ###############################################################################
   # Licensed Materials - Property of IBM.
   # Copyright IBM Corporation 2017. All Rights Reserved.
   # U.S. Government Users Restricted Rights - Use, duplication or disclosure
   # restricted by GSA ADP Schedule Contract with IBM Corp.
   #
   # Contributors:
   #  IBM Corporation - initial API and implementation
   ###############################################################################
     apiVersion: v1
     kind: Secret
     metadata:
       name: my-cam-secure-values-secret
       namespace: cp4aiops
   ```

```
    labels:
      app: cam
  data:
    httpProxy: aHR0cDovL3VzZXJuYW1lOnBhc3N3b3JkQHByb3h5X2hvc3Q6cHJveHlfcG9ydA==
    httpsProxy: aHR0cDovL3VzZXJuYW1lOnBhc3N3b3JkQHByb3h5X2hvc3Q6cHJveHlfcG9ydA==
    noProxy:
    mongoDbPassword: ZkZDWGI1WjlSNlg3TGhr
    mongoDbUrl: bW9uZ29kYjovL2NhbXVzZXI6ZkZDWGI1WjlSNlg3TGhrQGNhbS1tb25nbzoyNzAxNy9jYW0/c3NsPXRydWU=
    encryptionPassword: ZkZDWGI1WjlSNlg3TGhr
    contentGitHubTokens:
```

Note: If there are some environments, such as Git, vCenter, or OpenStack, that you need to access by bypassing the proxy, then add their IP address or domain names as the `noProxy` parameter value. Separate the values by using commas. **Note**: You must not leave any parameter value as blank. Overwrite the values that you want to change, and leave the other values as they are.

4. Save the YAML file.

5. Create the secret.

   ```
   oc create -f my-cam-secure-values-secret.yaml
   ```

   You can also create the secret by passing the parameter values at command line. You do not need to encode the values. The base64 conversion is automatically done when you run the command. See the following example:

   ```
   oc -n cp4aiops create secret generic my-cam-secure-values-secret --from-
   literal=httpProxy=http://username:password@proxy_host:proxy_port --from-
   literal=httpsProxy=http://username:password@proxy_host:proxy_port --from-literal=noProxy= --from-
   literal=mongoDbPassword=fFCXb5Z9R6X7Lhk  --from-
   literal=mongoDbUrl=mongodb://camuser:fFCXb5Z9R6X7Lhk@cam-mongo:27017/cam?ssl=true --from-
   literal=encryptionPassword=fFCXb5Z9R6X7Lhk --from-literal=contentGitHubTokens=
   ```

Change the secret during installation

Once you create the new secret with your secret values, use this new secret as a value for `secureValues.secretName` in the `IAConfig` resource specification. For more information on how to create `IAConfig` resource, see Online install of Infrastructure Automation for IBM Cloud Pak for AIOps or Online installation of Infrastructure Automation.

The following sample yaml shows `IAConfig` resource specification with customized `secureValues.secretName`.

```
cat << EOF | oc apply -f -
kind: IAConfig
apiVersion: aiops.ibm.com/v1alpha1
metadata:
  name: ibm-ia-installer
  namespace: cp4aiops
spec:
  imagePullSecret: ibm-entitlement-key
  infraAutoComposableComponents:
    - enabled: <Set true to install Infrastructure Management component of
{{site.data.keyword.infra_automation}} false otherwise>
      name: ibm-management-im-install
      spec: {}
    - enabled: <Set true to install Managed services component of {{site.data.keyword.infra_automation}}
false otherwise>
      name: ibm-management-cam-install
      spec:
        manageservice:
          secureValues:
            secretName: my-cam-secure-values-secret
  license:
    accept: <Set true to accept the license>
  storageClass: <Storage Class name that supports RWX>
  storageClassLargeBlock: <Select a storage class with a large block size (for example, 64k)>
EOF
```

Change the secret post installation

If you installed Managed services by using the default secret, then you need to replace the default secret with a new secret. First, create a new secret with your new secret values. Then set the name of the new secret as the value of `secureValues.secretName` in the `ManageService` custom resource either by using the Red Hat OpenShift Container Platform console or by the following OC command line

```
oc edit manageservice cam -n cp4aiops
```

Once you change the secret name in `secureValues.secretName`, save the configuration. The deployments that use the secret will be restarted automatically.

Change the value of secret data in your secret

If you want to change secret data value in the secret you created, then edit the secret you created and change the value. You must restart all the cam running pods that use the secret for the changes to take effect.

Configuring Managed services to use a proxy

If your corporate policies dictate that you use a proxy server for all HTTP(s) communication to the internet, then you must configure the Managed services to use that proxy server.

There are two steps required in order to enable Managed services to use the proxy server:

1. Create the secret and specify the http(s) proxy using the procedure explained in this topic, Using pre-created secrets in Managed services.

   Note: If there are some environments, such as Git, vCenter, or OpenStack, that you need to access by bypassing the proxy, then add their IP addresses or domain names as the **noProxy** parameter value while creating a secret. Separate the values by using commas.

2. Set **useProxy=true** in the installation parameters. For more information, see Installation parameters.

Configuring Managed services to import templates and services by using your github.com token

You can use your organization's github.com token to import the default provided terraform templates and services into Managed services library. These templates and services are imported into Managed services library during installation of the product.

In the previous releases, the default provided terraform templates and services were imported into Managed services library by using IBM provided token. The downside of this approach is if the IBM provided token is expired or deleted, then you must use the UI to change the token one at a time or contact IBM Support to change the token.

To avoid this downside, it is recommended that you use your organization's github.com token to import the out of box terraform templates and services into Managed services library instead of IBM provided token. Since the templates are stored in github.com, you can genereate new tokens by logging into github.com and then navigate to **Setting** > **Developer settings** > **Personal access tokens**.

contentGitHubTokens secret data

A secret data **contentGitHubTokens** is introduced to the list of pre-created secret data. For more information, see Using pre-created secrets in Managed services. The value for this new secret data **contentGitHubTokens** is a base64 encoded comma-separated list of github.com tokens. The following example shows you how to convert a list of GitHub tokens into base64 encoded string.

Example:

If you want to use the following token list as value for **contentGitHubTokens** secret data:

```
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, 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
```

then encode the above token list by using the following command"

```
echo "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, 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" | tr -d '\n' | base64 -w0
```

Execution of the above command results in the following base64 encoded string:

```
Z2hwX0V4SEptazhlU3k0R0VSWEc1U044T3JtTVBPTmdTUTNUU3gxWiwgZ2hwXzNUZ2FuUGFWcnVseW9ENEZvRU9JVjh5Ylp0dmU5eDBGSW5QUQ==
```

Use this base64 encoded string as the value of secret data **contentGitHubTokens**.

Procedure to use your base64 encoded token list

Once you get the base64 encoded token list, follow the procedure in Using pre-created secrets in Managed services to change the default IBM provided token.

Note

Do the following if your organizations token used in contentGitHubTokens is expired or revoked:

1. You must first regenerate a new base64 encoded token list.

2. Change the value for contentGitHubTokens in the secret you created by using the procedure in Using pre-created secrets in Managed services.

3. Since contentGitHubTokens secret data value is set as the value of the environment variable CONTENT_GITHUB_TOKENS in cam-iaas deployment, you must restart the cam-iaas pod for the changes to reflect.

Configuring Managed services to set the SSH keys and known hosts in provider terraform pod

To download and execute any terraform module that is stored in a repository and protected by a private key, you need to set private SSH keys and known hosts under **/home/terraform/.ssh** folder in the **cam-provider-terraform** pod. The **/home/terraform** folder in provider terraform is a read only directory hence you cannot create any files or folder under the directory. The following process helps you to create the private keys and known host files that are required to download and execute the external terraform modules.

Create secret that contains private keys and known hosts

You must first create a secret that contains the content of your private key and known host file. You can use the following command to create it.

```
    oc -n <namespace> create secret generic <secret_name> --from-file=id_rsa=<path-to-private-ssh-key-file> --from-file=known_hosts=<<path-to-knownhosts-file>
```

Alternately, you can define the secret in a yaml file **mysshkey.yaml** as follows:

```
    ###############################################################################
    # Licensed Materials - Property of IBM.
    # Copyright IBM Corporation 2022. All Rights Reserved.
    # U.S. Government Users Restricted Rights - Use, duplication or disclosure
    # restricted by GSA ADP Schedule Contract with IBM Corp.
    #
    # Contributors:
    #  IBM Corporation - initial API and implementation
    ###############################################################################
      apiVersion: v1
      kind: Secret
      metadata:
        name: <secret_name>
        namespace: <namespace>
      data:
        id_rsa: <base64 encoded private key content>
        known_hosts: <base64 encoded known hosts content>
      type: Opaque
```

Execute the following command to create the secret:

```
    oc create -f mysshkey.yaml
```

Procedure to add the key files to provider terraform pod

Once the secret is created, you can add the key and known host files to the **cam-provider-terraform** pod either during installation of Infrastructure Automation or after installation.

- To add the key files during installation, you must set the customization parameters under **spec.manageservice** section in the install yaml as follows:

```
cat << EOF | oc apply -f -
kind: IAConfig
apiVersion: aiops.ibm.com/v1alpha1
metadata:
  name: ibm-ia-installer
  namespace: cp4aiops
spec:
  imagePullSecret: ibm-entitlement-key
  infraAutoComposableComponents:
  - enabled: <Set true to install Infrastructure Management component of
{{site.data.keyword.infra_automation}} false otherwise>
    name: ibm-management-im-install
    spec: {}
  - enabled: <Set true to install Managed services component of {{site.data.keyword.infra_automation}}
false otherwise>
    name: ibm-management-cam-install
    spec:
      manageservice:
```

```
        camProviderTerraform:
          terraformSSHDir:
            terraformSSHDirSecret: <ssh_secret_name>
license:
  accept: <Set true to accept the license>
storageClass: <Storage Class name that supports RWX>
storageClassLargeBlock: <Select a storage class with a large block size (for example, 64k)>
EOF
```

- To set the key files after installation, do the following:

1. In Red Hat OpenShift Container Platform console, navigate to your installed **ManagedService** custom resource YAML.
2. Find the string **terraformSSHDirSecret** in the YAML.
3. Set the secret name of your ssh key and known host file as the value for **terraformSSHDirSecret**. If your secret name is **terraform-ssh-secret**, then your modified YAML looks as follows:

```
camProviderTerraform:
  isolateRuntime: true
  replicaCount: 1
  runtime:
    replicaCount: 1
  terraformSSHDir:
    terraformSSHDirSecret: terraform-ssh-secret
```

```
4. Save the changes.

5. Your `cam-provider-terraform` pod is restarted. To verify the `cam-provider-terraform` pod changes, you
can list the key files using `ls /home/terraform/.ssh`. This should display the ssh key and known host files.

**Note:** After you change the contents of the secret, ensure to restart the `cam-provider-terraform` pod to
reflect the changes.
```

- Provisioning Virtual Machines and Instances
- Infrastructure Automation Lifecycle
  - Provisioning
  - Provisioning Requests
    - Requirements for Provisioning Virtual Machines and Instances
    - Requirements for Provisioning Virtual Machines from Red Hat Virtualization Manager
    - PXE Provisioning
      - Connecting to a PXE Server
      - Creating System Image Types for PXE
      - Setting the PXE Image Type for a PXE Image
    - ISO Provisioning
      - Adding an ISO Datastore
      - Refreshing an ISO Datastore
      - Creating System Image Types for ISO
      - Setting the Image Type for an ISO Image
    - Customization Templates for Virtual Machine and Instance Provisioning
    - Customization Script Additions for Virtual Machine and Instance Provisioning
    - Adding a Customization Template
    - Provisioning Virtual Machines
      - Provisioning a Virtual Machine from a Template
      - Provisioning a Virtual Machine using Cloud-Init via REST API
      - Cloning a Virtual Machine
      - Publishing a Virtual Machine to a Template (VMware Virtual Machines Only)
      - Renaming a Provisioned Virtual Machine (VMware Virtual Machines Only)
    - Provisioning Instances
      - Provisioning an EC2 Instance from an Image
      - Provisioning an OpenStack Instance from an Image, Volume or Volume Snapshot
      - Provisioning an IBM Power Systems Virtual Servers Instance from an Image
      - Customizing Provisioning Dialogs
        - Adding a Provision Dialog for All Users
        - Creating a Custom Provision Dialog
      - Provisioning Profiles
        - Creating a Provisioning Profile Instance
        - Setting Provisioning Scope Tags
      - Managing Key Pairs
  - Working with Requests

- Provisioning Request Approval Methods
- Working with Provisioning Requests
  - Reloading the Status of Provisioning Requests
  - Approving a Provisioning Request
  - Denying a Provisioning Request
  - Copying a Provisioning Request
  - Editing a Provisioning Request
  - Deleting a Provisioning Request
  - Automatically Approving Requests
    - Enabling Global Defaults for Automatic Approval
    - Template Specific Approval Defaults
    - Assigning Tags to a Template for Auto Approval
    - Setting Provisioning Notification Email Addresses

- Fulfilling Requests
  - Fulfilling a Request
  - Default Execution Steps in States Instance
    - Quotas
      - Applying User or Group Quotas
      - Using Tags for Owner and Group Quotas
      - Applying a Tag to a User or User Group
      - State Machines
      - Customizing Provisioning States
      - Editing the Default State Instance
      - Viewing the Status of a Provisioning Request
      - Viewing a Provisioned Virtual Machine or Instance
      - Viewing a Virtual Machine or Instance Summary

- Catalogs and Services
  - Generic Objects
    - Viewing Generic Objects Classes
    - Creating Generic Objects Classes
    - Editing Generic Object Classes
    - Removing Generic Objects Classes
    - Exporting Generic Objects
    - Importing Generic Objects
  - Service Dialogs
    - Adding a Service Dialog
      - Creating a Service Dialog from a Container Template
      - Importing Service Dialogs
      - Exporting Service Dialogs
    - Methods
      - Creating a Method to Associate with the Dialog
      - Creating a Method in the Service Class
      - Creating an Instance in the Service Class
      - Associating a Method with an Automate Instance
  - Catalogs
    - Creating a Catalog Bundle
    - Copying a Catalog Bundle
    - Creating a Catalog Item
    - Copying a Catalog Item
    - Creating a Generic Catalog Item
    - Creating an Ansible Playbook Service Catalog Item
    - Creating an Ansible Tower Service Catalog Item
    - Creating an Amazon Service Catalog Item
    - Creating an Azure Service Catalog Item
    - Creating an OpenShift Template Catalog Item
    - Creating an Orchestration Catalog Item
    - Creating a VMware Content Library OVF Template Catalog Item
    - Provisioning a Service
  - Orchestration Stacks
    - Creating an Orchestration Template
    - Editing Orchestration Templates
    - Copying Orchestration Templates

        - Deleting Orchestration Templates
    - Retirement
        - Retiring Virtual Machines
            - Retiring Virtual Machines and Instances
            - Using the Console to Retire a Virtual Machine
            - Retiring a Virtual Machine Immediately
            - Setting a Retirement Date and Time for a Virtual Machine or Instance
            - Removing a Retirement Date for a Virtual Machine or Instance
        - Setting Retirement in a Provision Request
            - Scheduling Retirement in a Provision Request
        - Extending Retirement Dates
            - Creating a Custom Button to Extend Retirement
            - Changing the Number of Days to Extend Retirement

Provisioning Virtual Machines and Instances

Infrastructure Automation Lifecycle

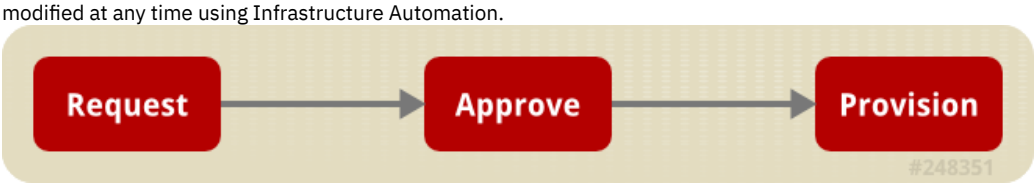
This guide discusses lifecycle activities such as provisioning and retirement that are part of the Infrastructure Automation Automate component. Infrastructure Automation Automate enables real-time, bidirectional process integration and adaptive automation for management events and administrative or operational activities.

- Operations Management with service level resource enforcement.
- Resource Management including datastore cleanup, snapshot aging and enforcement, and virtual machine or instance aging and retirement.
- Configuration and Change Management including enforced closed loop change management.
- Lifecycle Management such as provisioning, customization, reconfiguration, approval, CMDB updates, and retirement.

Provisioning requires the **Automation Engine** server role enabled. Check your server role settings in the settings menu, **Configuration > Server > Server Control**.

Provisioning

When a virtual machine or cloud instance is provisioned, it goes through multiple phases. First, the request must be made. The request includes ownership information, tags, virtual hardware requirements, the operating system, and any customization of the request. Second, the request must go through an approval phase, either automatic or manual. Finally, the request is executed. This part of provisioning consists of pre-processing and post-processing. Pre-processing acquires IP addresses for the user, creates CMDB instances, and creates the virtual machine or instance based on information in the request. Post-processing activates the CMDB instance and emails the user. The steps for provisioning may be modified at any time using Infrastructure Automation.



Provisioning Requests

The following options are available when making provisioning requests:

- Set an owner (User can do this using LDAP lookup)
- Assign a purpose (tag)
- Select a template or image from which to create a new virtual machine or instance respectively
- Choose placement
- Set hardware requirements
- Specify the vLan

- Customize the guest operating system
- Schedule the provisioning

Requirements for Provisioning Virtual Machines and Instances

Infrastructure Automation supports the provisioning of VMware ESX hypervisors. To provision a virtual machine from VMware providers, you must have an appliance with the Automation Engine role enabled.

If you are using a Windows template, the following configuration is required:

- To customize settings that are inside the operating system, Sysprep must be copied to the appropriate directory on your vCenter computer. Usually this location is: **`C:\Documents and Settings\All Users\Application Data\VMware\VMware VirtualCenter\sysprep`**. Copy the Sysprep tools to the relevant operating system subdirectory. If you are running a standard Win2008 operating system, this step is unnecessary as Sysprep is included as standard.
- The Windows template must have the latest version of VMware tools for its ESX Server. Check the VMware Site for more information. If you are creating a new password for the Administrator account, the Administrators password must be blank on the template. This is a limitation of Microsoft Sysprep.

See the VMware documentation for a complete list of customization requirements.

Requirements for Provisioning Virtual Machines from Red Hat Virtualization Manager

Item	Requirements
Red Hat Virtualization Manager version 4.0 or higher	Red Hat Virtualization Manager properly installed with API in default location https://server:8443/api
Red Hat Virtualization Manager History Database	Red Hat Virtualization Manager Data Warehouse (DWH) properly installed with access to the PostgreSQL database on the Red Hat Virtualization Manager server. Port 5432 open in iptables. md5 authentication allowed to Infrastructure Automation appliances in **`pg_hba.conf`**. PostgreSQL set to listen for connections on **`*:5432`** in **`postgresql.conf`**. Credentials provided during database setup to be used in Infrastructure Automation UI.
Storage Supported for Infrastructure Automation Virtual Machine Analysis	NFS - Infrastructure Automation server must be able to mount NFS storage domain. iSCSI / FCP - Cluster must use full Red Hat Enterprise Linux (not Red Hat Virtualization Hypervisor) Hosts. DirectLUN Hook installed on each host and registered to Red Hat Virtualization Managers. Must have Infrastructure Automation appliance in each Cluster with this storage type. Infrastructure Automation appliance virtual machine container must have DirectLUN attribute set. Local storage - Not yet supported (Red Hat does not recommend due to single point of failure).

PXE Provisioning

PXE is a boot method that allows you to load files from across a network link. Infrastructure Automation uses it for files required for provisioning virtual machines. PXE can be used for provisioning for either Red Hat Virtualization Manager or VMware.

Procedure Overview

1. Connect to the **PXE Server**.
2. Create a **System Image Type**.
3. Associate each **PXE** image with an image type.
4. Create a customization template.

Requirements for PXE Provisioning

- DHCP server configured with required PXE implementation
- PXE implementation for Linux virtual machine provisioning

- NFS or SAMBA read and write access to create and modify files on the PXE server
- Infrastructure Automation Server uses NFS mount to read and write the response files
- HTTP read access to the NFS share location as virtual machines use this URL to access PXE images and Kickstart or Cloud-Init configuration files
- Operating system installation media available to be streamed from PXE server
- Images configured for desired operating systems
- Kickstart or Cloud-Init templates to configure operating systems with desired packages

Additional Requirements for Provisioning Linux Virtual Machines

- Linux distribution kernel and ramdisk available over HTTP
- Linux sources available over HTTP
- Sample PXE menu item that boots this kernel

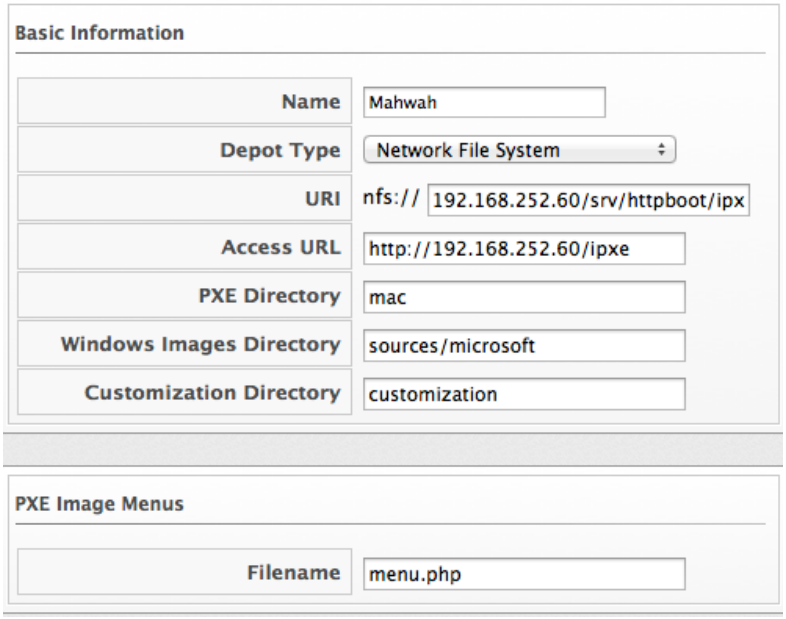
Additional Requirements for Provisioning Windows Virtual Machines

- WinPE ISO built with rhev-agent-tools (for RHEV-M environments) and configured to mount shares for Windows source files and Sysprep files and configured to run customization script
- Windows based WIM file with operating system installed and configured with Sysprep
- Sample Sysprep unattend file to be used with the operating system
- Sample PXE menu item that downloads WinPE ISO, mount in memdisk and boot into WinPE environment

Connecting to a PXE Server

The following procedure connects to a PXE server and adds its details to Infrastructure Automation.

1. Navigate to **Compute > Infrastructure > PXE**.
2. Click (**Configuration**), then (**Add a New PXE Server**).
3. In **Basic Information**, type a **Name** that will be meaningful in your environment.



4. For **Depot Type**, select either **Network File System** (NFS) or **Samba**. The fields to enter in the dialog depend on the **Depot Type**.
    - For NFS, type in the **URI**, **Access URL**, **PXE Directory**, **Windows Images Directory**, and **Customization Directory**. When you provision, Infrastructure Automation writes a text file to the **PXE Directory**. The file is named after the MAC address of the NIC that is assigned to the virtual machine. It contains where to get the kernel and initrd image. This file is removed after a successful provision. The **Windows Images Directory** is where the files are located on your NFS for the provisioning of Windows operating systems. The **Customization Directory** is where your Kickstart and Sysprep files are located.

- If using a **Depot Type** of **Samba**, you will not need **Access URL**, but you will need a **User ID**, and **Password**, in addition to the items required for NFS.

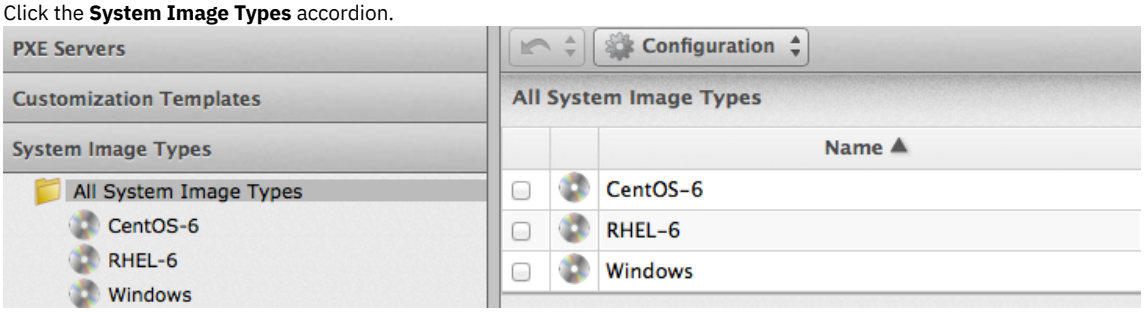
5. For **PXE Image Menus**, type the **Filename** for the PXE Boot menu.
6. Click **Add**.
7. Select the new PXE server from the tree on the left, and click (**Configuration**), then (**Refresh**) to see your existing images.

Next, create PXE Image types to associate with the customization templates and to specify if the image type is for a virtual machine, a host, or both.

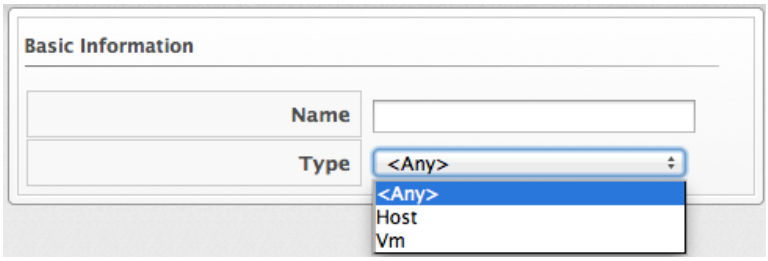
Creating System Image Types for PXE

The following procedure creates a system image type for PXE servers.

1. Navigate to **Compute > Infrastructure > PXE**.
2. Click the **System Image Types** accordion.



3. Click (**Configuration**), then (**Add a new System Image Type**).



4. In **Basic Information**, type in a **Name** and select a **Type**.
   - Use **Vm** if you want this image type to only apply to virtual machines.
5. Click **Add**.

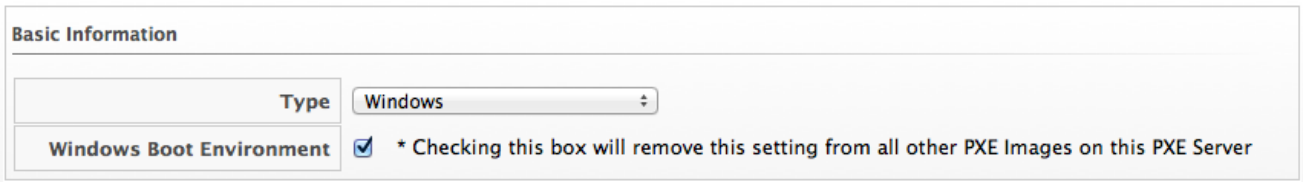
Example:

After creating the System Image Types, assign the types to each image on your PXE servers. To do this, you will select each image on the PXE server and identify its type.

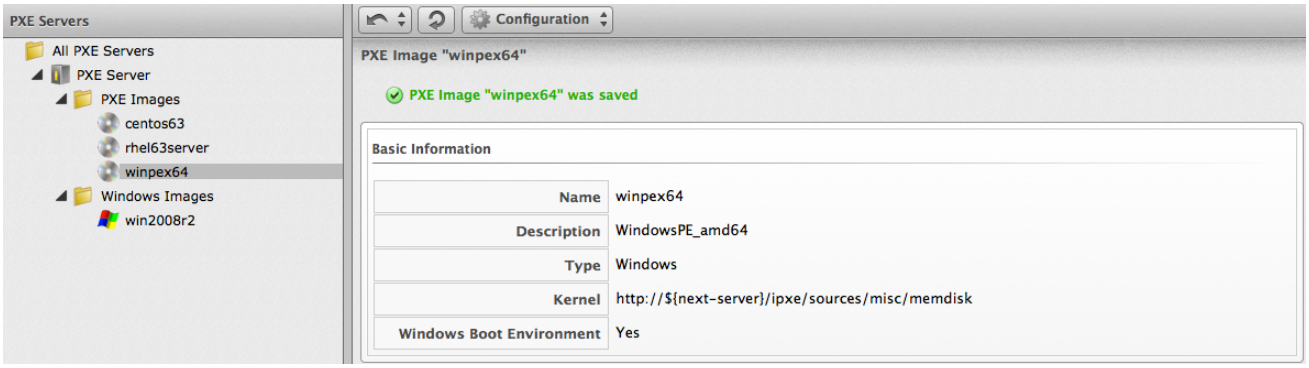
Setting the PXE Image Type for a PXE Image

The following procedure sets the image type for a chosen PXE image.

1. Navigate to **Compute > Infrastructure > PXE**.
2. Click the **PXE Servers** accordion and select the image that you want to set a type for.
3. Click (**Configuration**), then (**Edit this PXE Image**).
4. From the **Basic Information** area, select the correct type. If this PXE image will be used as the **Windows Boot Environment**, check **Windows Boot Environment**. At the time of this writing, only one PXE Image can be identified as the **Windows Boot Environment**. Therefore, checking one as the **Windows Boot Environment**, will remove that from any other PXE image with that check.



Click **Save**.



ISO Provisioning

Infrastructure Automation also allows ISO provisioning from Red Hat Virtualization Manager datastores. To use this feature, you will need to do the following before creating a provision request.

1. Add the **ISO Datastore**. The Red Hat Virtualization Manager system must have already been discovered or added into the VMDB. For more information, see Adding a Red Hat Enterprise Virtualization Manager Provider in *Managing Providers*.
2. Refresh the **ISO Datastore**.
3. Create a **System Image Type**.
4. Set the **ISO Image Type**.
5. **Create** a customization template.

Adding an ISO Datastore

The following procedure adds an ISO Datastore from your Red Hat Virtualization environment.

1. Navigate to **Compute > Infrastructure > PXE**.
2. Click the **ISO Datastores** accordion.
3. Click (**Configuration**), (**Add a new ISO Datastore**).
4. Select the Cloud or Infrastructure provider hosting the ISO Datastore.
5. Click **Add**.

The ISO datastore is added to Infrastructure Automation.

Refreshing an ISO Datastore

The following procedure refreshes the chosen ISO datastore and updates Infrastructure Automation with available ISOs.

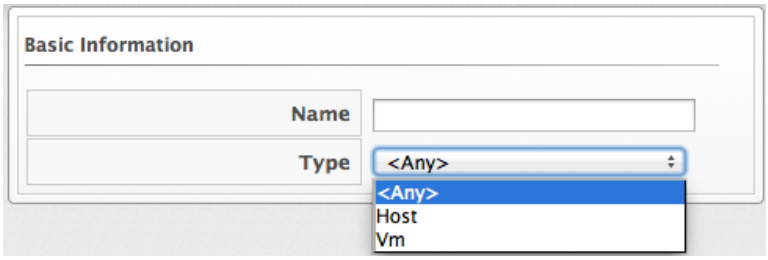
1. Navigate to **Compute > Infrastructure > PXE**.
2. Click the **ISO Datastores** accordion, and select an ISO datastore.
3. Click (**Configuration**), then click (**Refresh Relationships**).

Creating System Image Types for ISO

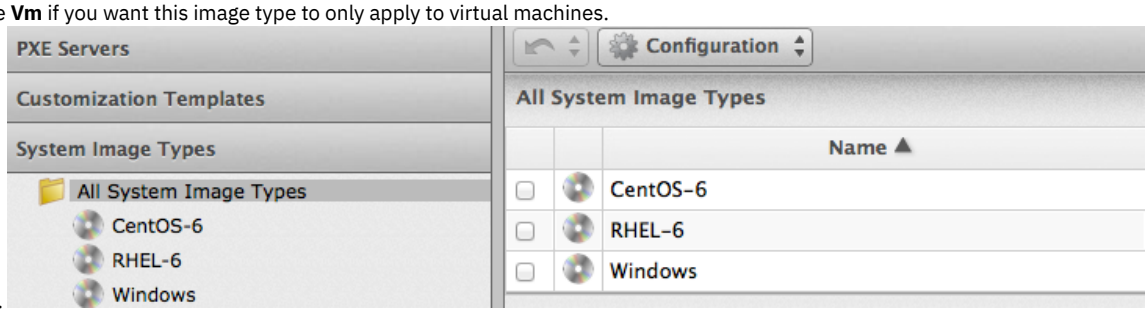
The following procedure creates a system image type for ISO Servers.

1. Navigate to **Compute > Infrastructure > PXE**.
2. Click the **System Image Types** accordion.
3. Click (**Configuration**), then (**Add a new System Image Type**).

4. In **Basic Information**, type in a **Name** and select a **Type**.



   - Use **Vm** if you want this image type to only apply to virtual machines.
5. Click **Add**.



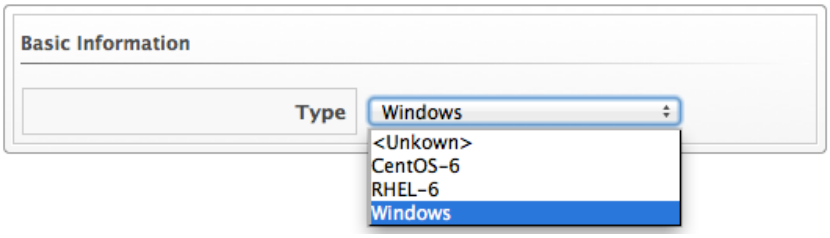
Example:

After creating the system image types, assign the types to each image on your ISO servers. To do this, you will select each image on the ISO server and identify its type.

Setting the Image Type for an ISO Image

The following procedure sets the image type for an ISO image.

1. Navigate to **Compute > Infrastructure > PXE**.
2. Click the **ISO Datastores** accordion, and select the image that you want to set a type for.
3. Click (**Configuration**), then (**Edit this ISO Image**).
4. From the **Basic Information** area, select the correct **Type**.



5. Click **Save**.

Customization Templates for Virtual Machine and Instance Provisioning

Add a customization template to provide **Kickstart**, **Cloud-Init**, or **Sysprep** files for the initial loading of the operating system.

- When creating a template using Red Hat Virtualization, install the **cloud-init** package on the source virtual machine. This enables Cloud-Init to source configuration scripts when a virtual machine built on that template boots.
- See Using Cloud-Init to Automate the Configuration of Virtual Machines in the Red Hat Virtualization *Administration Guide* for more information on using Cloud-Init in a Red Hat Virtualization environment.
- See the Cloud-Init Documentation web site for example scripts.
- The **Kickstart** file must be named **ks.cfg**.
- Set the new virtual machine to power down after provisioning is complete.
- Infrastructure Automation must use the virtual machine payload feature of Red Hat Virtualization to create a floppy disk containing the data from the selected customization template.
- Customize the installer to include the data written to the floppy disk payload.
- RHEL 7.5 and above

- **isolinux.cfg** NDASH add **ks=cdrom** to the **append** line
- **ks.cfg** NDASH which must minimally include:

```
### Pre Install Scripts
%pre

# Mount the floppy drive
modprobe floppy
mkdir /tmp/floppy
mount /dev/fd0 /tmp/floppy
%end

# Include ks.cfg file from the floppy (written by CFME based on selected customization template)
%include /tmp/floppy/ks.cfg
```

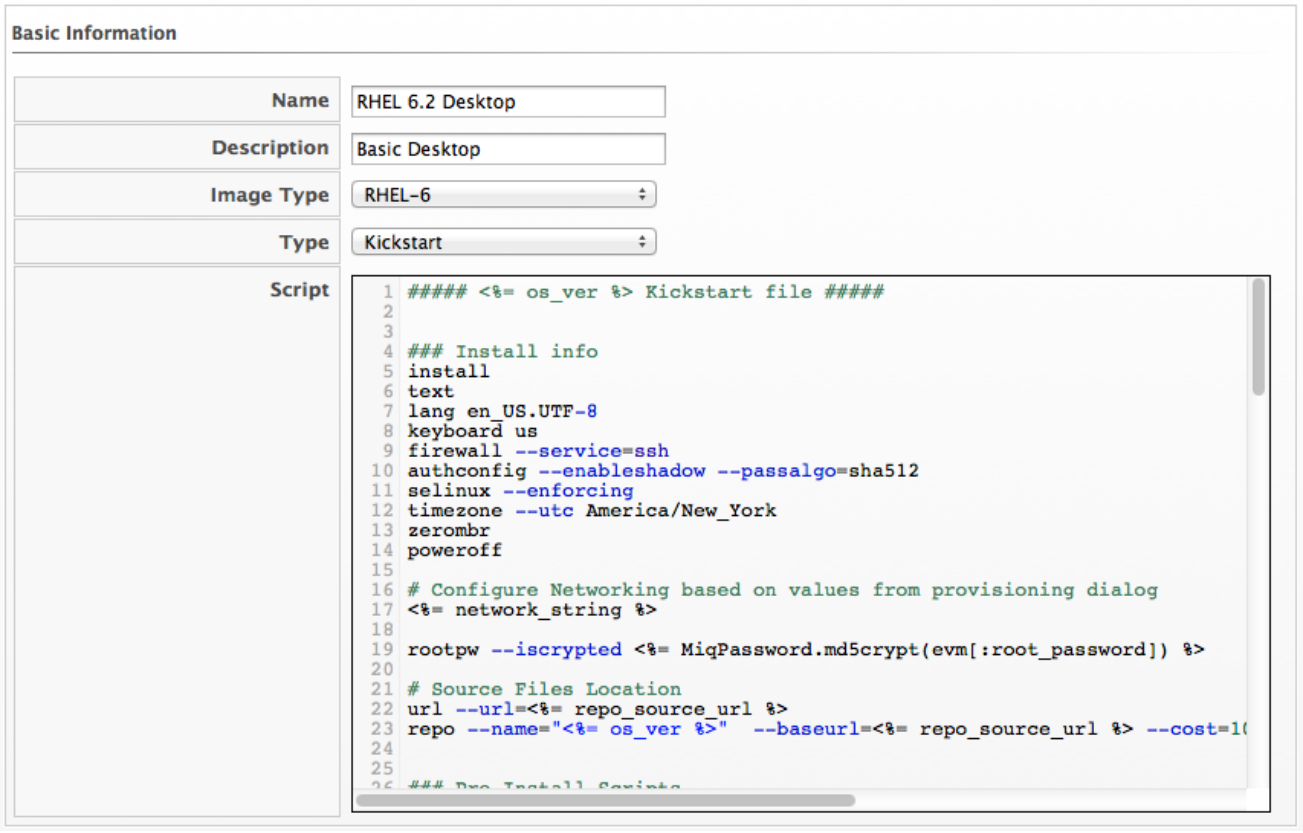
Customization Script Additions for Virtual Machine and Instance Provisioning

Customization Type	Reason to Include	Script entries		
Kickstart	Takes the values from the **Customize** tab in **Provisioning Dialog** and substitutes them into the script.	*Configure Networking based on values from provisioning dialog <% if evm[:addr_mode].first == 'static' %> <% network_string = "network –onboot yes –device=eth0 –bootproto=static –noipv6" %> <% ["ip", :ip_addr, "netmask", :subnet_mask, "gateway", :gateway, "hostname", :hostname, "nameserver", :dns_servers].each_slice(2) do	ks_key, evm_key	%> <% network_string << " –*{ks_key} # {evm[evm_key]} " unless evm[evm_key].blank? %> <% end %> <%= network_string %> <% else %> network –device=eth0 –bootproto=dhcp <% end %>
Kickstart	Encrypts the root password from the **Customize** tab in the **Provisioning Dialog**.	rootpw –iscrypted <%= ManageIQ::Password.md5crypt(evm[:root_password]) %>		
Kickstart	Sends status of the provision back to Infrastructure Automation Server for display in the Infrastructure Automation Console.			
Sysprep	Encrypts the root password from the **Customize** tab in the **Provisioning Dialog**. The value for the **AdministratorPassword** line must be inserted to use the password from the **Provision Dialog** and encrypt it.	<UserAccounts> <AdministratorPassword> <Value> <%= ManageIQ::Password.sysprep_crypt(evm[:root_password]) %></Value> <PlainText>false</PlainText> </AdministratorPassword> </UserAccounts>		

Adding a Customization Template

1. Navigate to **Compute > Infrastructure > PXE**.
2. Click the **Customization Templates** accordion.
3. Click (**Configuration**), (**Add a new Customization Template**).

4. In **Basic Information**, type in a **Name** and **Description**.



5. Select the **Image Type**. This list should include the PXE image types you created.
6. In **Type**, select **Kickstart** or **CloudInit** for Linux based systems, and **Sysprep** for Windows based system.
7. In the **Script** area, either paste the script from another source or type the script directly into the Infrastructure Automation interface.
8. Click **Add**.

Example:

The default dialogs show all possible parameters for provisioning. To limit the options shown, see Customizing Provisioning Dialogs.

Provisioning Virtual Machines

There are four types of provisioning requests available in Infrastructure Automation:

1. Provision a new virtual machine from a template
2. Clone a virtual machine
3. Publish a virtual machine to a template
4. Provision a virtual machine using cloud-init via REST API.

Provisioning a Virtual Machine from a Template

You can provision virtual machines through various methods. One method is to provision a virtual machine directly from a template stored on a provider.

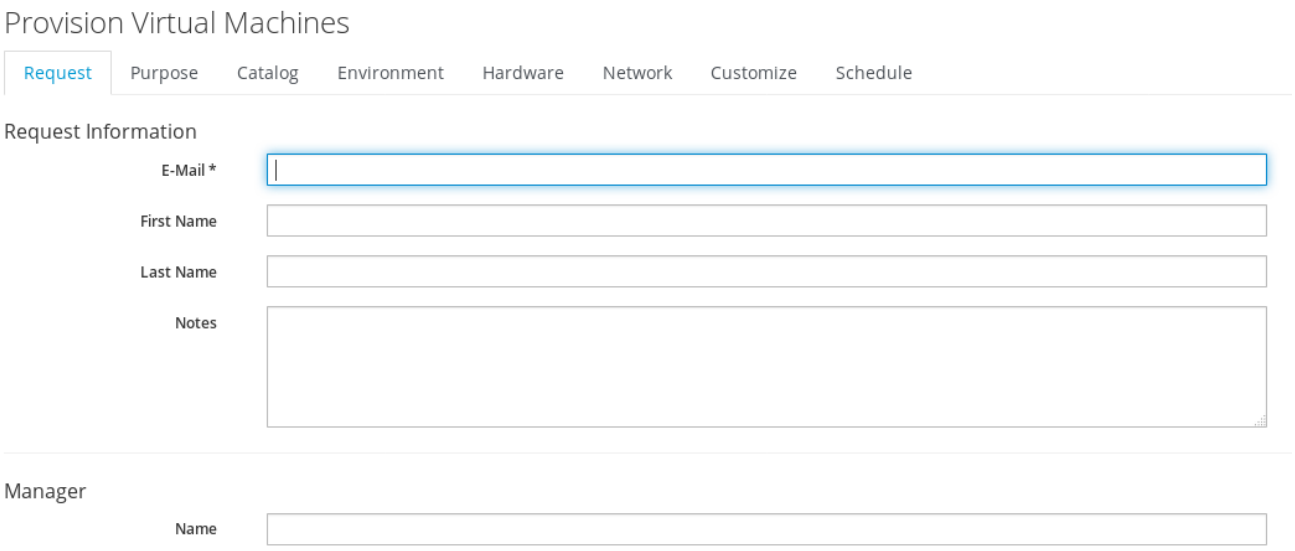
Important:

- To provision a virtual machine, you must have the “Automation Engine” role enabled.
- During virtual machine provisioning, the **Customize** tab is hidden if the template has an unknown operating system (OS) type. To make the **Customize** tab visible in the user interface, you will need to set the OS type from the provider or perform SmartState analysis on the template to detect the OS type.

To provision a virtual machine from a template:

1. Navigate to **Compute > Infrastructure > Virtual Machines**.

2. Click (**Lifecycle**), and then (**Provision VMs**).
3. Select a template from the list.
4. Click **Continue**.
5. On the **Request** tab, enter the request information.

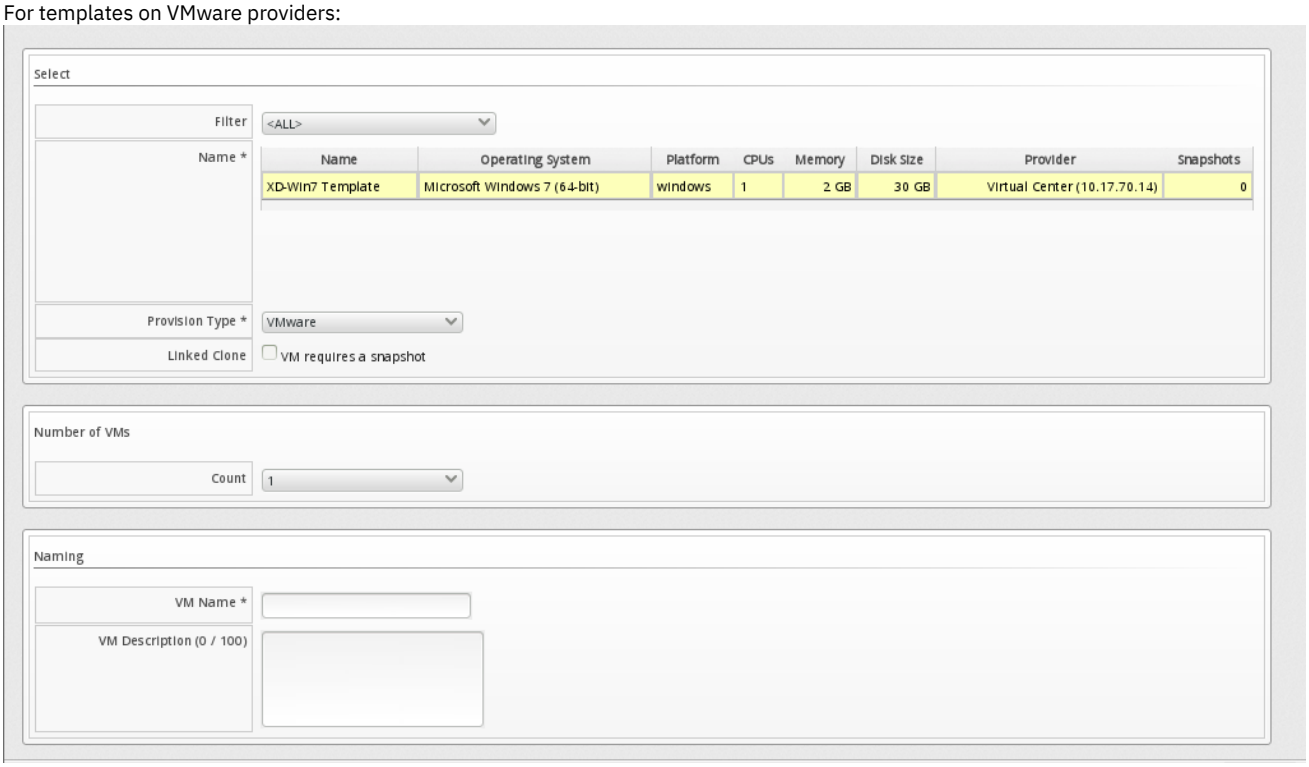


In **Request Information**, enter your name and email address. The requester will receive status emails about the provisioning request such as auto-approval, quota, provision complete, retirement, request pending approval, and request denied.

Note:

Parameters with a * next to the label are required to submit the provisioning request. To change the required parameters, see Customizing Provisioning Dialogs.

6. Click the **Purpose** tab to select the appropriate tags for the provisioned virtual machines.
7. Click the **Catalog** tab to select the template to provision from. This tab is context sensitive based on provider.
8. For templates on VMware providers:



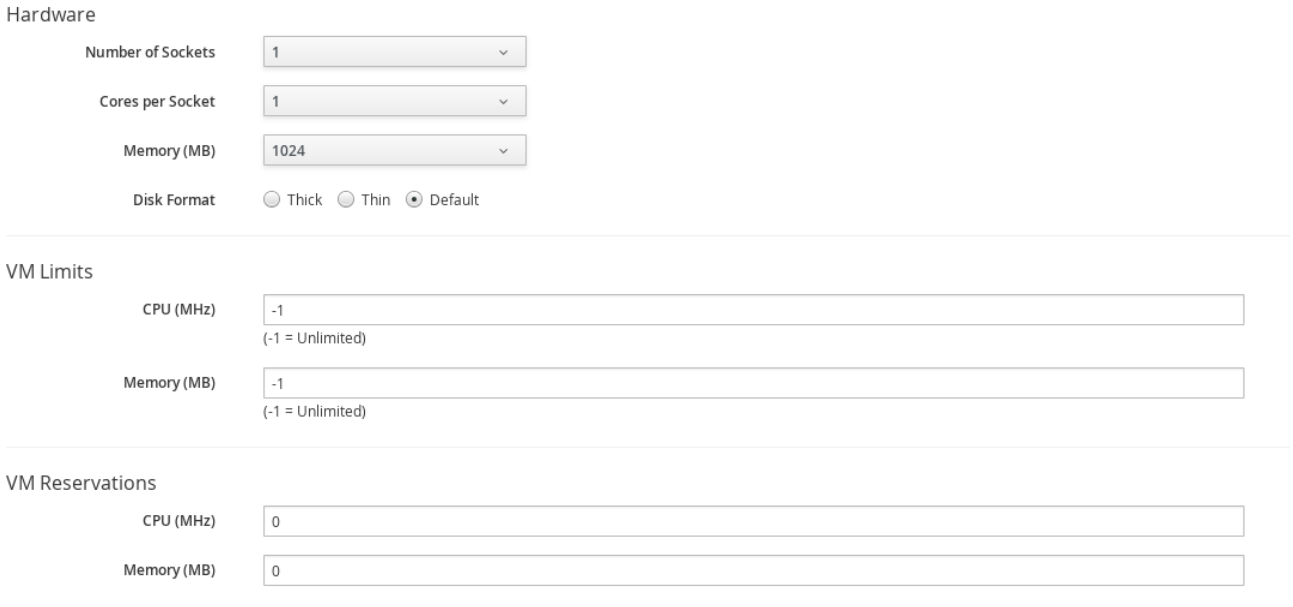
1. For **Provision Type**, select **VMware** or **PXE**.

1. If **VMware** is selected, select **Linked Clone** to create a linked clone to the virtual machine instead of a full clone. Since a snapshot is required to create a linked clone, this box is only enabled if a snapshot is present. Select the snapshot you want to use for the linked clone.
2. If **PXE** is selected, select a PXE **Server** and **Image** to use for provisioning.

2. Under **Count**, select the number of virtual machines to create in this request.
3. Use **Naming** to specify a virtual machine name and virtual machine description. When provisioning multiple virtual machines, a number will be appended to the virtual machine name.

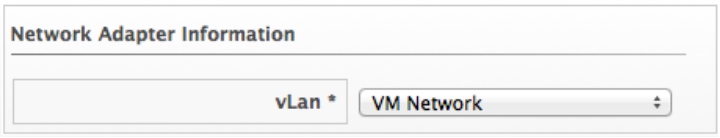
9. For templates on Red Hat providers:
    1. Select the **Name** of a template to use.
    2. For **Provision Type**, select either **ISO**, **PXE**, or **Native Clone**. You must select **Native Clone** in order to use a Cloud-Init template.
        1. If **Native Clone** is selected, select **Linked Clone** to create a linked clone to the virtual machine instead of a full clone. This is equivalent to *Thin Template Provisioning* in Red Hat Virtualization. Since a snapshot is required to create a linked clone, this box is only enabled if a snapshot is present. Select the snapshot to use for the linked clone.
        2. If **ISO** is selected, select an ISO **Image** to use for provisioning.
        3. If **PXE** is selected, select a PXE **Server** and **Image** to use for provisioning.
    3. Under **Count**, select the number of virtual machines you want to create in this request.
    4. Use **Naming** to specify a **VM Name** and **VM Description**. When provisioning multiple virtual machines, a number will be appended to the **VM Name**.
10. Click the **Environment** tab to decide where you want the new virtual machines to reside.
    1. If provisioning from a template on VMware, you can either let Infrastructure Automation decide for you by checking **Choose Automatically**, or select a specific cluster, resource pool, folder, host, and datastore. VMware virtual machines can also be provisioned to a clustered datastore by selecting it under **Datastore**. Additionally, you can assign a storage profile to a VMware virtual machine under **Datastore** to configure the virtual machine to operate using a storage profile from that datastore.

       Note, read-only datastores are excluded when provisioning a virtual machine.
    2. If provisioning from a template on Red Hat, you can either let Infrastructure Automation decide for you by checking **Choose Automatically**, or select a datacenter, cluster, host and datastore.
11. Click the **Hardware** tab to set hardware options.



    1. In **Hardware**, set the number of sockets, cores per socket, memory in MB, and disk format: thin, pre-allocated/thick or same as the provisioning template (default).
    2. For VMware provisioning, set the **VM Limits** of CPU and memory the virtual machine can use.
    3. For VMware provisioning, set the **VM Reservation** amount of CPU and memory.

12. Click **Network** to set the vLan adapter. Additional networking settings that are internal to the operating system appear on the **Customize** tab.

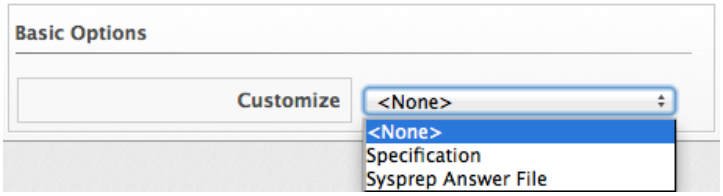


    1. In **Network Adapter Information**, select the **vLan**.

        Note:

        A VMware virtual machine can be provisioned to a DVPortgroup by selecting it from the **vLan** list. Prior to provisioning a virtual machine, the DVPortgroup must be created on a vSphere Distributed Switch (VDS) in VMware vCenter in order for Infrastructure Automation to list the DVPortgroup under **vLan**.

13. Click **Customize** to customize the operating system of the new virtual machine. These options vary based on the operating system of the template.

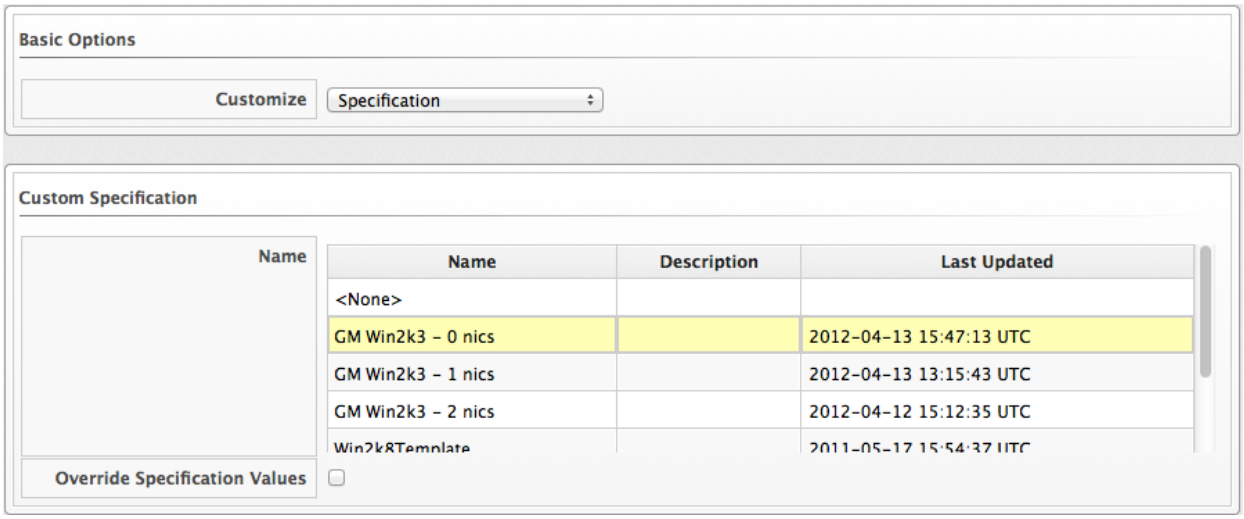


14. For Windows provisioning:

    1. To use a custom specification from the provider, click **Specification**. To select an appropriate template, choose from the list in the custom specification area. The values that are honored by Infrastructure Automation display.

        Note:

        Any values in the specification that do not show in the Infrastructure Automation console's request dialogs are not used by Infrastructure Automation. For example, for Windows operating systems, if you have any run once values in the specification, they are not used in creating the new virtual machines. Currently, for a Windows operating system, Infrastructure Automation honors the unattended GUI, identification, workgroup information, user data, windows options, and server license. If more than one network card is specified, only the first is used.



        To modify the specification, select **Override Specification Values**.

    2. Select **Sysprep Answer File**, to upload a Sysprep file or use one that exists for a custom specification on the Provider where the template resides. To upload a file, click **Browse** to find the file, and then upload. To use an answer file in **Customization Specification**, click on the item. The answer file will automatically upload for viewing. You cannot make modifications to it.

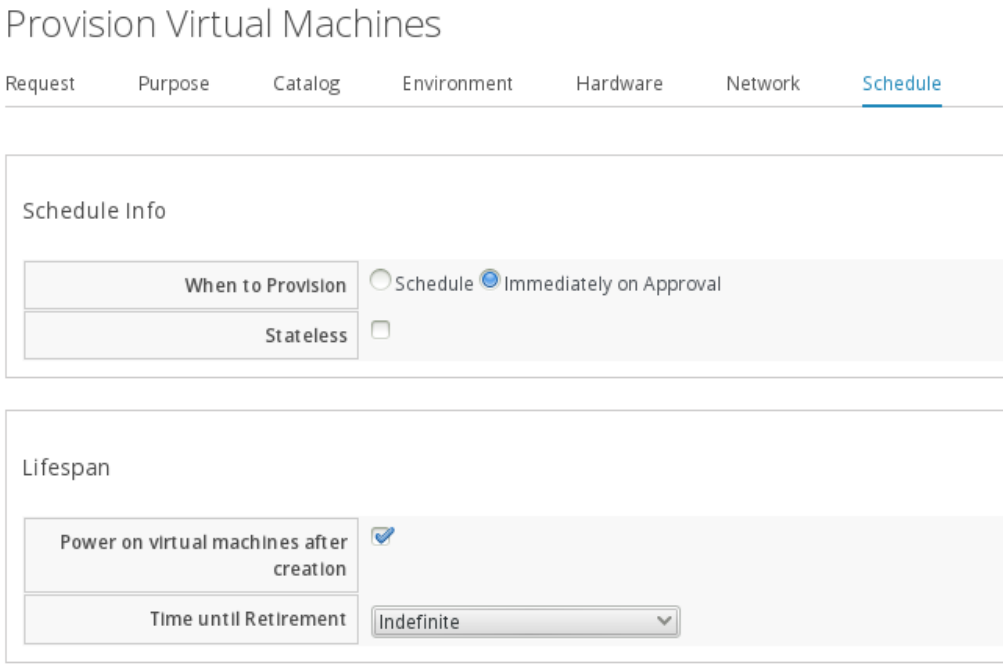
15. For Linux provisioning:

    1. Under **Credentials**, enter a **Root Password** for the **root** user to access the instance.
    2. Enter a **IP Address Information** for the instance. Leave as **DHCP** for automatic IP assignment from the provider.
    3. Enter any **DNS** information for the instance if necessary.
    4. Select **Customize Template** for additional instance configuration. Select from the Kickstart or Cloud-Init customization templates stored on your appliance.

16. Click the **Schedule** tab to select when provisioning begins.

    1. In **Schedule Info**, select when to start provisioning. If you select **Schedule**, you will be prompted to enter a date and time. Select **Stateless** if you do not want the files deleted after the provision completes. A stateless provision does not write to the disk so it requires the PXE files on the next boot.

    2. In **Lifespan**, select to power on the virtual machines after they are created, and to set a retirement date. If you select a retirement period, you will be prompted for when you want a retirement warning.



17. Click **Submit**.

The provisioning request is sent for approval. For the provisioning to begin, a user with the administrator, approver, or super administrator account role must approve the request. The administrator and super administrator roles can also edit, delete, and deny the requests. You will be able to see all provisioning requests where you are either the requester or the approver.

After submission, the appliance assigns each provision request a **Request ID**. If an error occurs during the approval or provisioning process, use this ID to locate the request in the appliance logs. The Request ID consists of the region associated with the request followed by the request number. As regions define a range of one trillion database IDs, this number can be several digits long.

Request ID Format

Request 99 in region 123 results in Request ID 123000000000099.

Provisioning a Virtual Machine using Cloud-Init via REST API

Cloud-init is a tool for automating the initial setup of virtual machines. In Infrastructure Automation, you can use cloud-init via REST API to provision a virtual machine that was created based on a template.

Note:

To use cloud-init, the template from which the virtual machine is provisioned must have cloud-init package installed, and have the **Use Cloud-Init/Sysprep** option selected.

For a virtual machine provision request via REST API, ensure the following two fields in the request's body are set correctly, otherwise cloud-init may not work.

- VLAN
- sysprep_enabled

VLAN.

The value of **VLAN** in the API request can be one of the following options:

VLAN value	Note	vNIC profile
<empty>	Must be written within the <..>	No vNIC profile
<Template>	Must be written within the <..>	Use the template's default vNIC profile

VLAN value	Note	vNIC profile
profile_name (network_name)	Must include a space between profile_name (network_name), otherwise it will fail. For example, **ovirtmgmt (ovirtmgmt)**	Set the specified vNIC profile.
vNIC profile ID		Set the specified vNIC profile, such as `3a8dce01-dd59-4a46-8a6f-823acccda79f`.

sysprep_enabled.

The value of sysprep_enabled in the API request must be in the following format.

Note:

Sysprep_enabled is written in the order: [value] - [it's meaning] - [how it appears in the Infrastructure Automation user interface virtual machine provision dialog's **Customize** tab]

For Windows template:

- **"fields"** - Sysprep Specification In the Infrastructure Automation user interface, when you navigate to the virtual machine provision dialog (**Compute > Infrastructure

  Virtual Machine**), this option located under the **Customize** tab's **Customize** drop-down list is called **Sysprep Specification**.

- **"file"** - Sysprep answer file in the Infrastructure Automation user interface, when you navigate to the virtual machine provision dialog (**Compute > Infrastructure > Virtual Machine**), this option located under the **Customize** tab's **Customize** drop-down list is called **Sysprep answer file**.

For Linux template:

- **"fields"** - Customized template and any Customized parameters will be used In the Infrastructure Automation user interface, when you navigate to the virtual machine provision dialog (**Compute > Infrastructure

  Virtual Machine**), this option located under the **Customize** tab's **Customize** drop-down list is called **Specification**.

Note:

For cloud-init to work *(that is, to have the provisioned virtual machine marked with **Use cloud-init** and the customized template as well as customized parameters, if any, will be used)* "sysprep_enabled" must be set to "fields". If you do not set it correctly, the customized template will be ignored and the provisioned virtual machine will not be marked with **Use cloud-init**; although, the template from which the virtual machine is provisioned has it marked.

For both Windows and Linux template:

- **"disabled"** - Do not customize In the Infrastructure Automation user interface, when you navigate to the virtual machine provision dialog (**Compute > Infrastructure > Virtual Machine**), this option located under the **Customize** tab's **Customize** drop-down list is called **<None>**. The customized template will be ignored, and the provisioned virtual machine will not be marked with **Use cloud-init**, even though the template from which the virtual machine was provisioned, has it marked. The default value is 'disabled'.

Note:

For an example of virtual machine provisioning request using cloud-init via REST API, see *Provision Requests* in the Infrastructure Automation API Guide.

Cloning a Virtual Machine

Virtual machines can be cloned in other providers as well.

1. Navigate to **Compute > Infrastructure > Virtual Machines**, and select the virtual machine you want to clone.
2. Click (**Lifecycle**), and then (**Clone selected item**).
3. Enter the requested information in the dialogs. Be sure to check the **Catalog** tab.
4. Schedule the request on the **Schedule** tab.
5. Click **Submit**.

Publishing a Virtual Machine to a Template (VMware Virtual Machines Only)

1. Navigate to **Compute > Infrastructure > Virtual Machines**, and select the virtual machine you want to publish as a template.
2. Click (**Lifecycle**), and then (**Publish selected VM to a Template**).

3. Enter the requested information in the dialogs. Be sure to check the **Catalog** tab.
4. Schedule the request on the **Schedule** tab.
5. Click **Submit**.

Renaming a Provisioned Virtual Machine (VMware Virtual Machines Only)

Infrastructure Automation allows you to rename a VMware virtual machine without having to reprovision it.

To rename a VMware virtual machine:

1. Navigate to **Compute > Infrastructure > Virtual Machines**, and select the VMware virtual machine you want to rename.
2. Click (**Configuration**), (**Rename selected item**).
3. In the **Basic Information** screen, provide a new **name**.
4. Click **Save**.

The renamed virtual machine will appear in the inventory view.

Provisioning Instances

Cloud instances follow the same process (Request, Approval, Deployment) as a standard virtual machine from virtualization infrastructure. First, a user makes a request for instances and specifies the image, volume or volume snapshot, tags, availability zone and hardware profile flavor. Second, the request goes through the approval phase. Finally, Infrastructure Automation executes the request.

Provisioning an EC2 Instance from an Image

1. Navigate to **Compute > Clouds > Instances**.
2. Click (**Lifecycle**), then click (**Provision Instances**).
3. Select an image from the list presented.
4. Click **Continue**.
5. On the **Request** tab, enter information about this provisioning request. In **Request Information**, type in at least a first and last name and an email address. This email is used to send the requester status emails during the provisioning process for items such as auto-approval, quota, provision complete, retirement, request pending approval, and request denied. The other information is optional. If the Infrastructure Automation Server is configured to use LDAP, you can use the **Look Up** button to populate the other fields based on the email address.

   Note:

   Parameters with a * next to the label are required to submit the provisioning request. To change the required parameters, see Customizing Provisioning Dialogs.
6. Click the **Purpose** tab to select the appropriate tags for the provisioned instance.
7. Click the **Catalog** tab for basic instance options.
   1. To change the image to use as a basis for the instance, select it from the list of images.
   2. Select the **Number of VMs** to provision.
   3. Type a **VM Name** and **VM Description**.
8. Click the **Environment** tab to select the instance's **Availability Zone**, **Virtual Private Cloud**, **Cloud Subnet**, **Security Groups**, and **Elastic IP Address**. If no specific availability zone is required, select the **Choose Automatically** checkbox.
9. Click the **Properties** tab to set provider options such as hardware flavor and security settings.
   1. Select a flavor from the **Instance Type** list.
   2. Select a **Guest Access Key Pair** for access to the instance.
   3. Select the **CloudWatch** monitoring level. Leave as **Basic** for the default EC2 monitoring.
10. Click the **Customize** tab to set additional instance options.
    1. Under **Credentials**, enter a **Root Password** for the **root** user access to the instance.
    2. Enter a **IP Address Information** for the instance. Leave as **DHCP** for automatic IP assignment from the provider.

3. Enter any **DNS** information for the instance if necessary.
4. Select a **Customize Template** for additional instance configuration. Select from the Cloud-Init scripts stored on your appliance.

11. Click the **Schedule** tab to set the provisioning and retirement date and time.
    1. In **Schedule Info**, choose whether the provisioning begins upon approval, or at a specific time. If you select **Schedule**, you will be prompted to enter a date and time.
    2. In **Lifespan**, select whether to power on the instances after they are created, and whether to set a retirement date. If you select a retirement period, you will be prompted for when to receive a retirement warning.
12. Click **Submit**.

The provisioning request is sent for approval. For the provisioning to begin, a user with the admin, approver, or super admin account role must approve the request. The admin and super admin roles can also edit, delete, and deny the requests. You will be able to see all provisioning requests where you are either the requester or the approver.

After submission, the appliance assigns each provision request a **Request ID**. If an error occurs during the approval or provisioning process, use this ID to locate the request in the appliance logs. The Request ID consists of the region associated with the request followed by the request number. As regions define a range of one trillion database IDs, this number can be several digits long.

Request ID Format

Request 99 in region 123 results in Request ID 123000000000099.

Provisioning an OpenStack Instance from an Image, Volume or Volume Snapshot

Create a request to provision Red Hat OpenStack Platform cloud instances from images, volumes, and volume snapshots using Infrastructure Automation. Only bootable volumes not in use will be available.

1. Browse to menu: **Compute > Clouds > Instances**
2. Click (**Lifecycle**), then click (**Provision Instances**).
3. Select an OpenStack image, volume or volume snapshot from the list presented. These files must be available on your OpenStack provider.
4. Click **Continue**.
5. On the **Request** tab, enter information about this provisioning request. In **Request Information**, type in at least an email address. This email is used to send the requester status emails during the provisioning process for items such as auto-approval, quota, provision complete, retirement, request pending approval, and request denied. The other information is optional. If the Infrastructure Automation Server is configured to use LDAP, you can use the **Look Up** button to populate the other fields based on the email address.

   Note:

   Parameters with a * next to the label are required to submit the provisioning request. To change the required parameters, see Customizing Provisioning Dialogs.
6. Click the **Purpose** tab to select the appropriate tags for the provisioned instance.
7. Click the **Catalog** tab for basic instance options.
   1. To change the source file to use as a basis for the instance, select it from the list of images, volumes, or volume snapshots.
   2. Select the **Number of Instances** to provision.
   3. Type a **Instance Name** and **Instance Description**.
8. Click the **Environment** tab to select the instance's **Cloud Tenant**, **Availabilty Zones**, **Cloud Network**, **Security Groups**, and **Public IP Address**. If no specific Cloud Tenant is required, select the **Choose Automatically** checkbox.
9. Click the **Properties** tab to set provider options such as flavors and security settings.
   1. Select a flavor from the **Instance Type** list.
   2. Select a **Guest Access Key Pair** for access to the instance. For more information about key pairs, see Managing Key Pairs.
10. Click the **Volumes** tab to provision any volumes with the instance. Volumes are useful for augmenting ephemeral storage of instances with persistent, general-purpose block storage:
    1. Fill in the **Volume Name** and **Size (gigabytes)** fields.
    2. If you want the volume to be deleted once the instance terminates (thereby making it non-persistent), check **Delete on Instance Terminate**.

3. To provision and add multiple volumes to the instance, click **Add Volume**. Doing so will add new fields you can fill in.

   For more information about persistent storage in OpenStack, see the Red Hat OpenStack Platform *Storage Guide*.

11. Click the **Customize** tab to set additional instance options.
    1. Under **Credentials**, enter a **Root Password** for the **root** user access to the instance.
    2. Enter a **IP Address Information** for the instance. Leave as **DHCP** for automatic IP assignment from the provider.
    3. Enter any **DNS** information for the instance if necessary.
    4. Select a **Customize Template** for additional instance configuration. Select from the Cloud-Init scripts stored on your appliance.
12. Click the **Schedule** tab to set the provisioning and retirement date and time.
    1. In **Schedule Info**, choose whether the provisioning begins upon approval, or at a specific time. If you select **Schedule**, you will be prompted to enter a date and time.
    2. In **Lifespan**, select whether to power on the instances after they are created, and whether to set a retirement date. If you select a retirement period, you will be prompted for when to receive a retirement warning.
13. Click **Submit**.

The provisioning request is sent for approval. For the provisioning to begin, a user with the admin, approver, or super admin account role must approve the request. The admin and super admin roles can also edit, delete, and deny the requests. You will be able to see all provisioning requests where you are either the requester or the approver.

After submission, the appliance assigns each provision request a **Request ID**. If an error occurs during the approval or provisioning process, use this ID to locate the request in the appliance logs. The Request ID consists of the region associated with the request followed by the request number. As regions define a range of one trillion database IDs, this number can be several digits long.

Request ID Format

Request 99 in region 123 results in Request ID 123000000000099.

Provisioning an IBM Power Systems Virtual Servers Instance from an Image

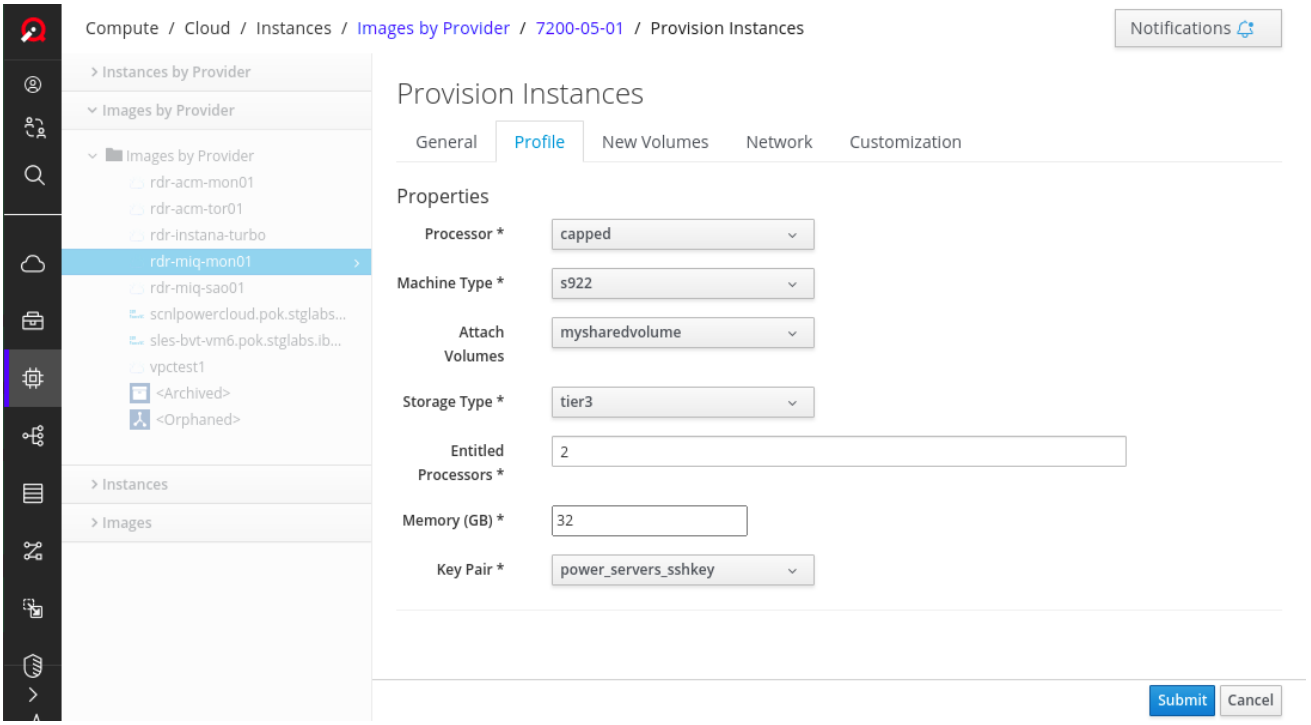
For an overview of IBM Power Systems Virtual Servers, including terminology used in Infrastructure Automation, see Getting started with IBM Power Systems Virtual Servers

Create a request to provision IBM Power Systems Virtual Servers instances from images.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click **Lifecycle**, then click (**Provision Instances**).
3. Select an image from the list presented. Each image is associated with a single Power Systems Virtual Servers provider instance. If there are multiple provider instances registered be sure to select an image associated with the desired instance.

   Notes:

   - Images must already be imported into a Power Systems Virtual Servers provider instance before becoming available in the Infrastructure Automation inventory. This can be done using the IBM Cloud web interface, CLI, or API. For detailed instructions see IBM Cloud PowerVS docs: Importing a boot image and IBM Cloud CLI: ibmcloud pi image-create.
   - SAP HANA on IBM Power Systems Virtual Servers requires special SLES for SAP HANA images. These boot images are added to a Power Systems Virtual Servers provider instance automatically when provisioning an SAP HANA instance through the IBM Cloud web interface. Once the SLES for SAP HANA image has been added, subsequent instances can be provisioned using Infrastructure Automation.
4. Click **Continue**.
5. Click the **General** tab for general instance options.
   1. Select the **Number of Instances** to provision.
   2. Type an **Instance Name**.
6. Click the **Profile** tab for hardware profile options.



1. Select the **Processor** type.

   Note: See Power Systems Virtual Servers documentation for more information.

2. Select a **Machine Type**.

   Note: When provisioning with an SAP HANA image, the **Machine Type** field is used to select the SAP profile. The SAP profile includes the number of entitled processors and memory (these fields will be automatically set and can only be changed by selected a different SAP profile).

3. Optionally, select **Attach Volumes** items.

   Note: A VM cannot have disks from different storage types. Only existing volumes matching the **Storage Type** will be available for selection.

4. Select a **Storage Type**.

   Note: See Power Systems Virtual Servers documentation for more information on volume types.

5. Enter number of **Entitled Processors**.

   Notes:

   - If 'dedicated' **Processor** is selected, **Entitled Processors** must be allocated in whole numbers.
   - If 'shared' or 'uncapped' **Processor** is selected, **Entitled Processors** must be allocated in a multiple of '0.25'.
   - When provisioning with an SAP HANA image this field is set by the profile (**Machine Type**) selection.

6. Enter **Memory (GB)**.

   Note: When provisioning with an SAP HANA image this field is set by the profile (**Machine Type**) selection.

7. Optionally, select a **Key Pair**.

7. Click the **New Volumes** tab to optionally create new volumes to be attached to the VM.

   All new volumes will be the **Storage Type** selected in the **Profile** tab. For each new volume:

   1. Type a volume **Name**.
   2. Enter **Size (GB)**.
   3. Select a **Sharable**.

8. Click the **Network** tab to select networks.

1. Select an existing network from **Network Interface**. If 'None' is selected then **Attach to New Public Network** must be selected.
2. Select **Attach to New Public Network** to create a new public network to attach to the VM.

9. Click the **Customization** tab for additional options.
    1. Use the **Upload File** section to upload a Cloud-init file.
    2. In the **Other** section, select **Migratable** and **VM pinning**

       Note: Learn more about VM pinning.
    3. In the **IP Address Information** section, optionally enter an IP address to request for the selected **Network Interface** in the **Network** tab.
10. Click **Submit**.

The provisioning request is sent for approval. For the provisioning to begin, a user with the admin, approver, or super admin account role must approve the request. The admin and super admin roles can also edit, delete, and deny the requests. You will be able to see all provisioning requests where you are either the requester or the approver.

After submission, the appliance assigns each provision request a **Request ID**. If an error occurs during the approval or provisioning process, use this ID to locate the request in the appliance logs. The Request ID consists of the region associated with the request followed by the request number. As regions define a range of one trillion database IDs, this number can be several digits long.

Request ID Format

Request 99 in region 123 results in Request ID 123000000000099.

Customizing Provisioning Dialogs

The default set of provisioning dialogs shows all possible options. However, Infrastructure Automation also provides the ability to customize which tabs and fields are shown. You can decide what fields are required to submit the provisioning request or set default values.

For each type of provisioning, there is a dialog that can be created to adjust what options are presented. While samples are provided containing all possible fields for provisioning, you can remove what fields are shown but cannot add new fields or tabs.

Edit the dialogs to:

1. Hide or show provisioning tabs.
2. Hide or show fields. If you hide an attribute, the default will be used, unless you specify otherwise.
3. Set default values for a field.
4. Specify if a field is required to submit the request.
5. Create custom dialogs for specific users.

Adding a Provision Dialog for All Users

1. Login to the Infrastructure Automation console for the Infrastructure Automation server where you want to change the dialog.
2. Navigate to **Automation > Embedded Automate > Customization**.
3. Click the **Provisioning Dialogs** accordion.
4. Click the type of dialog you want to create: **Host Provision**, **VM Provision** or **VM Migrate**.
5. Select one of the default dialogs.
6. Click **Configuration**, and then click **(Copy this Dialog)**.
7. Type a new **Name** and **Description** for the dialog.
8. In the **Content** field,
    - To remove a tab from display, change its display value to ignore. By choosing ignore, you not only hide the tab, but also skip any fields on that tab that were required. To show the tab, change the display value to show.
    - To hide a field, change its `:display:` value from `:edit` to `:hide`. To display fields of most data types, use `:edit`. To display a button, use `:show`. To set a default value for a field, add `:default: defaultvalue` to the list of parameters for the field. Set the `:required:` parameter to either `true` or `false` based on your needs.

      Note:

If you set `:required:` to `true`, the field must have a value for the provision request to be submitted.

9. Click **Add**.

If you are using **Provisioning Profiles**, you can specify a specific file that holds the customizations. To do this, you must create an instance mapping to this file in the Infrastructure Automation **Applications/provisioning/profile/VM provisioning by group** class. By default, if you are using provisioning profiles and the group does not have a defined instance, the appropriate default dialog file will be used based on the type of provisioning selected.

Creating a Custom Provision Dialog

1. Navigate to **Automation > Embedded Automate > Customization**.
2. Click on the **Provisioning Dialogs** accordion.
3. Click on the type of dialog you want to create, **Host Provision**, **VM Provision** or **VM Migrate**.
4. Select one of the default dialogs.
5. Click (**Configuration**), and then (**Copy this Dialog**).
6. Rename the dialog as shown in the examples below.

Type of Provision	Dialog Name
Provision Virtual Machine from a template	miq_provision_dialogs_groupname_template Example: miq_provision_dialogs_ EvmGroup-user_self_service _template
Clone a Virtual Machine	miq_provision_dialogs_groupname_clone_to_vm Example: miq_provision_dialogs_ EvmGroup-user_self_service _clone_to_vm
Publish a Virtual Machine to a template	miq_provision_dialogs_groupname_clone_to_template Example: miq_provision_dialogs_ EvmGroup-user_self_service _clone_to_template

7. Make any changes you need.
8. In the **Content** field,
   - To remove a tab from display, change its display value to ignore. By choosing ignore, you not only hide the tab, but also skip any fields on that tab that were required. To show the tab, change the display value to show.
   - To hide a field, change its `:display:` value from `:edit` to `:hide`. To ensure the field does not get turned back on by a workflow model, use `:display_override: :hide`. To display fields of most data types, use `:edit`. To display a button, use `:show`. To set a default value for a field, add `:default:`
     `defaultvalue` to the list of parameters for the field. Set the `:required:` parameter to either `true` or `false` based on your needs.

     Note:

     If you set `:required:` to `true`, the field must have a value for the provision request to be submitted.
9. Click **Add**.

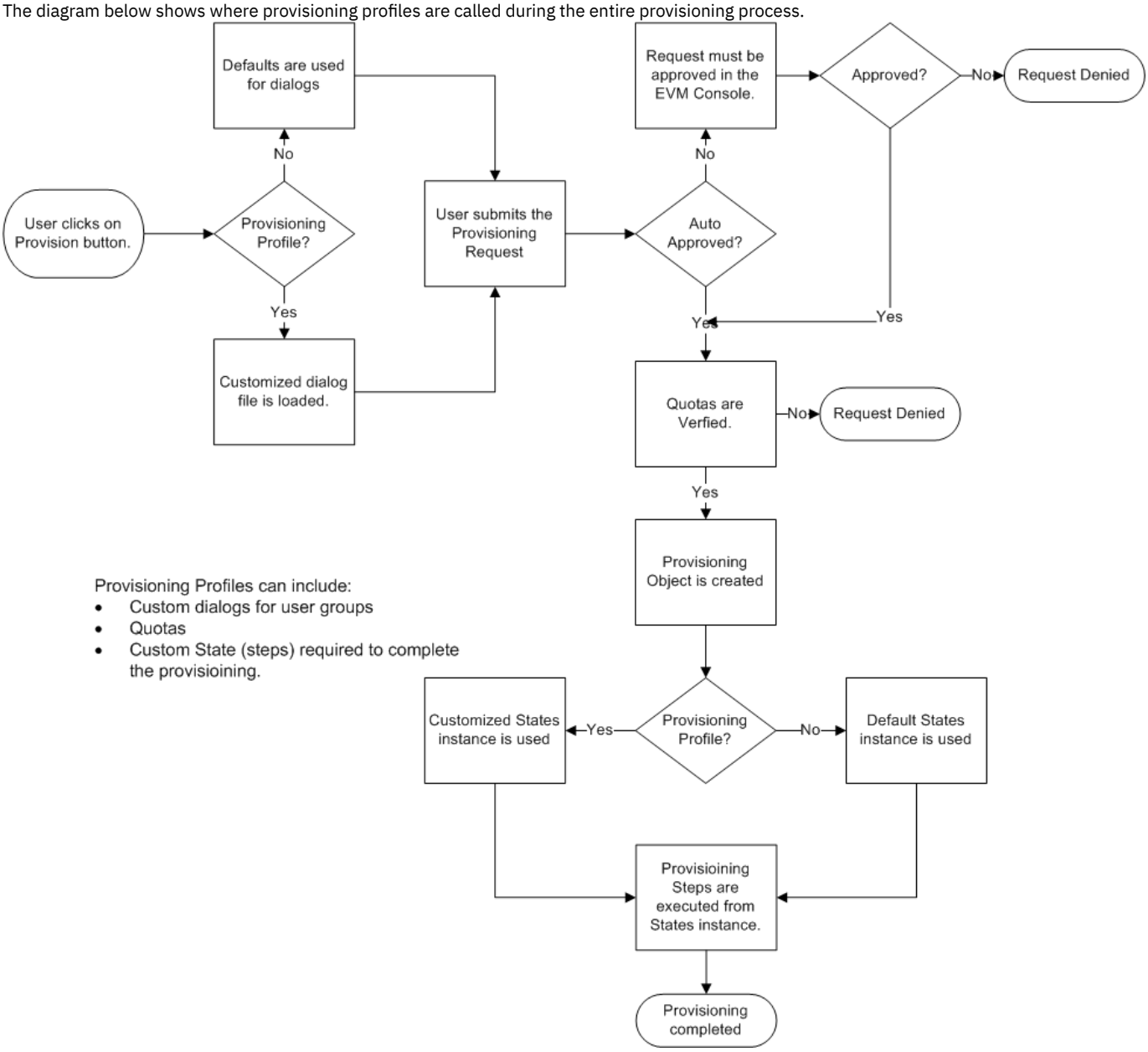
Enter the name of the new dialog into the dialog name field in the appropriate Infrastructure Automation **Applications/provisioning/profile instance**. This dialog can now be referred to in an instance in the Provisioning Profiles class so that it can be used for groups of users.

Provisioning Profiles

Provisioning profiles can be used to customize the dialogs and the state machine (steps used to provision the machine). Profiles can be created for LDAP or Infrastructure Automation groups. To use provisioning profiles:

- Create a **Provisioning Profile** instance for the LDAP or Infrastructure Automation group. If no instance exists, then default settings will be used.
- If customizing dialogs, create a custom dialog file, and specify the name of that file in the provisioning profile instance. If customizing the states for provisioning, create a state instance and set the name of the state instance in the provisioning profile instance.

The diagram below shows where provisioning profiles are called during the entire provisioning process.



Creating a Provisioning Profile Instance

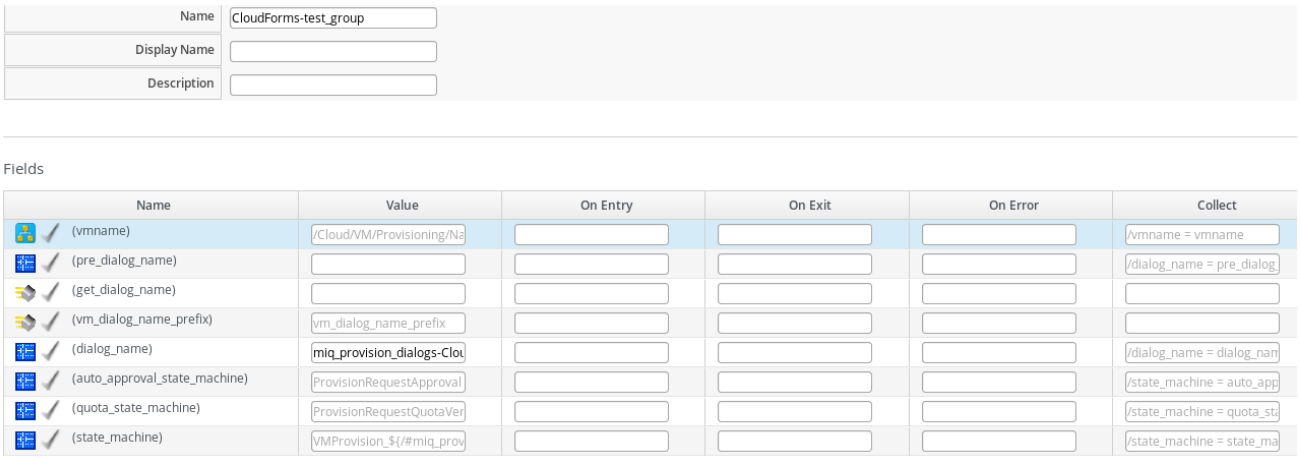
1. Navigate to **Automation > Embedded Automate > Explorer**.
2. Using the tree located in the accordion, click **DOMAIN > Cloud > VM > Provisioning > Profile**.

   Note:

   DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.

   This example uses the **Cloud** Namespace, but can also use the **Infrastructure** namespace.
3. Click (**Configuration**), (**Add a New Instance**).
4. Make the name of the tag identical to the name of the LDAP or Infrastructure Automation group you are creating the instance for, replacing spaces in the group name with underscores. For example, change **Infrastructure Automation-test group** to **Infrastructure Automation-**

test_group.



5. In the dialog name field, enter the name of the customized dialog file. This file must reside on the Infrastructure Automation appliance in the **`/var/www/miq/vmdb/db/fixtures`** directory. Red Hat recommends naming the file in the format **`miq_provision_dialogs-groupname.rb`** and copying this file to all Infrastructure Automation appliances. For instructions on creating a custom dialog file, see Customizing Provisioning Dialogs.

   Note:

   Be sure that the custom dialog file exists. If it does not, an error will appear when the user clicks on the **Provisioning** button in the Infrastructure Automation console.

6. Click **Add**.

Setting Provisioning Scope Tags

Some non-default placement methods, for example the **redhat_best_placement_with_scope** or **vmware_best_fit_with_scope** methods, may require you to set **Provisioning Scope** tags for a host and a datastore.

To enable these resources for all groups, set the scope to **All**. To limit access to a select group, create a tag in the **Provisioning Scope** category with the exact name of the user group and set this tag on the desired resources. See Tags in *General Configuration* for information on creating tags.

To set the scope for a datastore:

1. Navigate to **Compute > Infrastructure > Datastores**.
2. Select the datastore to set the provisioning scope for.
3. Click (**Policy**), and then (**Edit Tags**).
4. From the **Select a customer tag to assign** drop down, select **Provisioning Scope** and then a value for the tag from the next drop down menu.
5. Click **Save**.

Managing Key Pairs

Key pairs allow you to manage SSH access between a user and provisioned instance. For more information about key pairs in OpenStack, see Manage Key Pairs in the *Instances and Images Guide*.

To manage key pairs, browse to menu: **Compute > Clouds > Key Pairs**. From there, you can view a list of available key pairs. Click on a key pair to view its details.

To create a new key pair:

1. Browse to menu: **Compute > Clouds > Key Pairs**.
2. Click **Configuration**, (**Add a new Key Pair**).
3. Enter a **Name** for the key pair.
4. If you want to use a public key, copy its contents into the **Public Key (optional)** field.

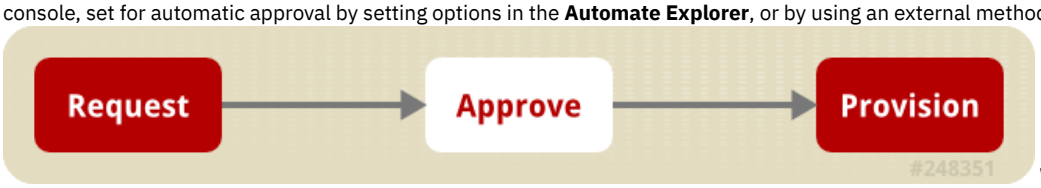
   Note: You must have a public key value for the IBM POWER VS provider to create the key pair. An error message is returned if you fail to add a public key value and can be viewed in **Settings > Tasks**.

5. Select which cloud provider on which to create the key pair. The key pair will then be available for use by instances in that provider.
6. Click **Add**.

Working with Requests

Provisioning Request Approval Methods

In this chapter, you will learn about the different approval methods. The request can be approved manually in the Infrastructure Automation console, set for automatic approval by setting options in the **Automate Explorer**, or by using an external method.



When using an external method, the approval actually takes place on the external system and is sent directly for execution. This chapter discusses how to view and edit requests in the Infrastructure Automation Console, how to approve a request, and how to set automatic approval parameters.

Working with Provisioning Requests

After a provisioning request is sent, if you have proper authority, you can copy, edit, delete, approve, or deny a request.

After submission, the appliance assigns each provision request a **Request ID**. If an error occurs during the approval or provisioning process, use this ID to locate the request in the appliance logs. The **Request ID** consists of the region associated with the request followed by the request number. As regions define a range of one trillion database IDs, this number can be several digits long.

Request ID Format

Request 99 in region 123 results in **Request ID 123000000000099**.

Reloading the Status of Provisioning Requests

1. Navigate to **Services > Requests**.
2. Click (**Refreshes the current display**).

Approving a Provisioning Request

After a user creates provisioning request, administrators have the ability to approve the request and allow Infrastructure Automation to complete virtual machine or instance creation.

1. Navigate to **Services > Requests**.
2. Click on the request you want to approve.
3. Click (**Approve this request**).
4. Type in a **Reason** for the approval.

Denying a Provisioning Request

1. Navigate to **Services > Requests**.
2. Click on the request you want to deny.
3. Click (**Deny this request**).
4. Type in a **Reason** for the denial.

Copying a Provisioning Request

1. Navigate to **Services > Requests**.
2. Click on the request you want to copy.
3. Click (**Copy original provision request**).
4. Make changes to the request.
5. Click **Submit**.

If the logged in user is not same as the requester or the request has been already approved or denied, you cannot edit or delete the request.

Editing a Provisioning Request

1. Navigate to **Services > Requests**.
2. Click on the request you want to edit.
3. Click (**Edit the original provision request**).
4. Make changes to the request.
5. Click **Submit**.

Deleting a Provisioning Request

1. Navigate to **Services > Requests**.
2. Click on the request you want to delete.
3. Click (**Delete this request**).
4. Click **OK** to confirm.

Automatically Approving Requests

You can set thresholds for automatic approval of provisioning requests and, therefore, remove the requirement to manually approve the request. You can do this either as a global default or on a per template basis.

Enabling Global Defaults for Automatic Approval

To enable a global set of default approval values, edit the defaults instance by navigating to **Automation > Embedded Automate > Explorer**, then **DOMAIN > Cloud|Infrastructure > VM > Provisioning > StateMachines > ProvisionRequestApproval** in the accordion menu. The parameters in this instance are used by the methods in that same class. By default, the maximum number of virtual machines or instances that can be automatically approved for provisioning is 1. To skip the check for the maximum number of virtual machines, set this field to 0. Set this field to -1 to force manual approval. At a minimum, you must change this parameter for all others to be validated.

1. Navigate to **Automation > Embedded Automate > Explorer**.
2. From the tree in the accordion menu, select **DOMAIN > Cloud > VM

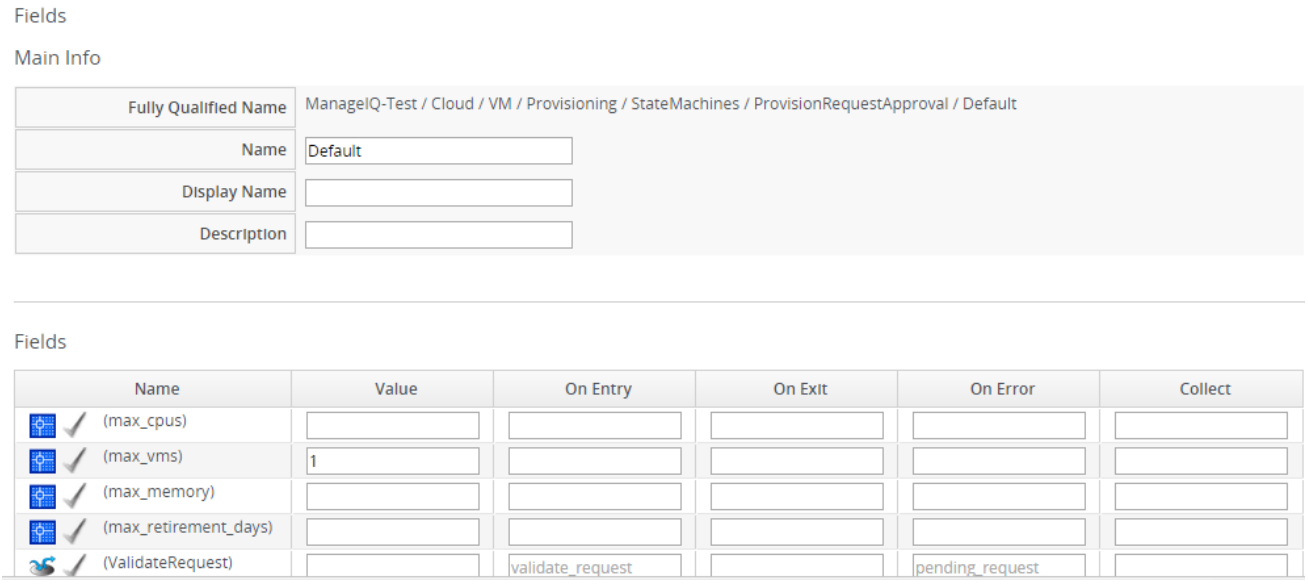
   Provisioning > StateMachines > ProvisionRequestApproval Class**.

   Note:

   DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.

   This example uses the **Cloud** Namespace but can also use the **Infrastructure** namespace.

3. Click (**Configuration**), then (**Edit this instance**).



Note:

Do not change any values other than those listed below. Doing so may prevent the automatic approval process from running.

- Use **max_cpus** to set the number of CPUs allowed to approve automatically the provisioning request.
- Use **max_vms** to set the maximum number of virtual machines or instances that are allowed to be provisioned automatically approve the request. If this is set to blank, no requests will be automatically approved.
- Use **max_memory** to set the maximum memory allowed to approve automatically the provisioning request.
- Use **max_retirement_days** to set the maximum number of days until the virtual machine or instance is retired to automatically approve this request.
- If a value is blank or **0**, the parameter is ignored.

4. Click **Save**.

The thresholds for automatic approval are set. The next time a provision request is created these thresholds will be checked. If the requirements are met, the provisioning request will be approved with no user intervention.

Template Specific Approval Defaults

Infrastructure Automation provides tags that can be used to set default automatic approval values on a per template or image basis. These values **supersede** those in the **Automate** model. Use these tags to eliminate the need for manual approval for all provisioning requests. To enable automatic approval, assign the tags directly to templates or images.

Category Display Name (Name)	Use (Sample values)
Auto Approve Max CPU (prov_max_cpus)	Sets the maximum number of CPUs that can be automatically approved in a single provisioning request. Sample Values: 1, 2, 3, 4, 5
Auto Approve Max Memory (prov_max_memory)	Sets the maximum number of memory that can be automatically approved in a single provisioning request. Sample Values: 1, 2, 4, 8 (in GB)
Auto Approve Max Retirement Days (prov_max_retirement_days)	Sets the maximum number of days until retirement that can be automatically approved in a single provisioning request. Sample Values: 30, 60, 90, 180 (in days)
Auto Approve Max VM (prov_max_vms)	Sets the maximum number of virtual machines or instances that can be automatically approved in a single provisioning request. Sample Values: 1, 2, 3, 4, 5

Assigning Tags to a Template for Auto Approval

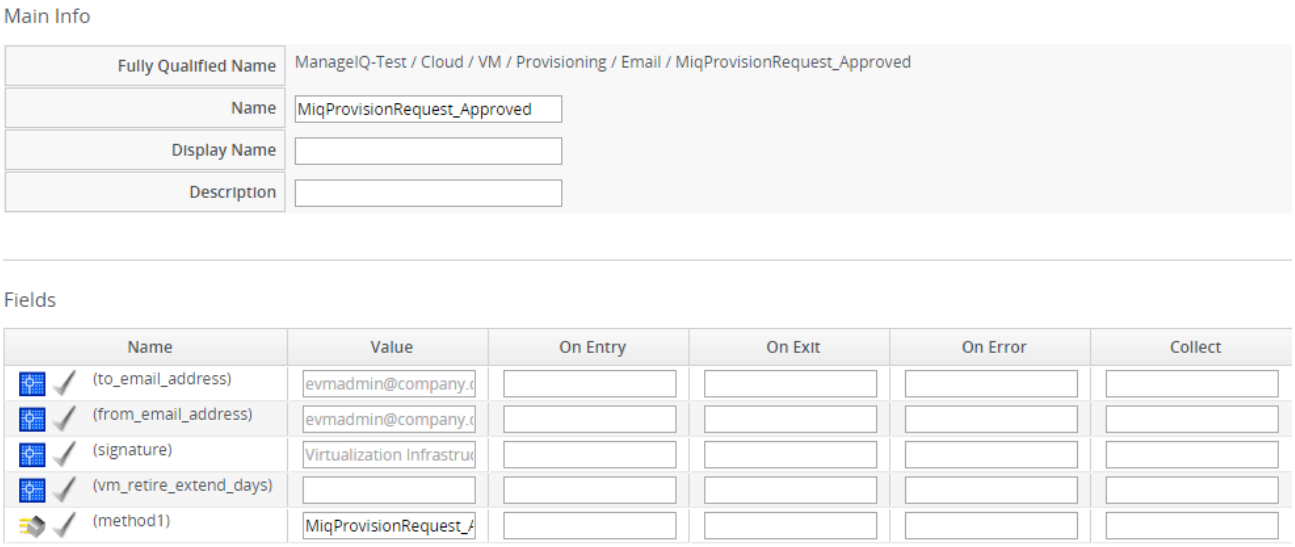
1. Navigate to **Compute > Infrastructure > Virtual Machines**.
2. Click the **Templates** accordion, and select the templates that you want to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. Select a customer tag from the first dropdown, and then a value for the tag.

The thresholds for automatic approval for a specific template are set. The next time a provision request is created for this template these thresholds will be checked. If the requirements are met, the provisioning request will be approved with no user intervention.

Setting Provisioning Notification Email Addresses

Infrastructure Automation contains a set of Automate instances for provisioning. These Automate instances also include email fields to set the sender and recipient of provisioning notifications, such as requests. These fields are set to **evmadmin@company.com** as a default.

1. Navigate to **Automation > Embedded Automate > Explorer**.
2. Select an instance within the chosen class.
3. Click (**Configuration**), then (**Edit this instance**).
4. Type the desired email addresses in the **to_email_address** and **from_email_address** fields.
5. Click **Save**.



Fulfilling Requests

Fulfilling a Request

After a request has been approved, Infrastructure Automation then goes through the steps required to complete the request. The steps followed for a regular provision from a virtual machine to a virtual machine (not to a template) are found by navigating to **Automation > Embedded Automate > Explorer**, then listed under **DOMAIN > Cloud|Infrastructure > VM > Provisioning > StateMachines > VMProvision_VM > Provision VM from Template (template)**. The value for each state shows where the instance resides in the **Datastore** accordion. The default set of execution steps is shown below. For more information on state machines, see State Machines.

Default Execution Steps in States Instance

Step	Description
Customize Request	Apply customizations.
Acquire IP Address	Integrates with IPAM (IP Address Management) to get an IP Address.
Acquire MAC Address	Integrates with IPAM to get a MAC Address.
Register DNS	Integrates with IPAM to register with DNS.
Register CMDB	Integrates with CMDB (Configuration Management Database) to register with the CMDB.
Register AD	Integrates with IPAM to register with active directory.
Placement	Determine placement of virtual machine or instance.
PreProvision	Pre-provisioning steps.
Provision	Create the virtual machine or instance.
CheckProvisioned	Check that the new virtual machine or instance is in the VMDB.
PostProvision	Post-provisioning steps.
Register DHCP	Integrate with IPAM to register the IP address with DHCP Server.
Activate CMDB	Integrate with IPAM to activate the virtual machine or instance in the CMDB.
Email owner	Send email to owner that the virtual machine or instance has been provisioned.

Quotas

Quotas allow you to establish maximum usage thresholds for an user, group, or tenant for provisioned virtual machines or instances and are integrated into provisioning profiles. These maximums are checked after the approval but before the actual provision request is started. The quota is set for the tenant or group as a whole.

Applying User or Group Quotas

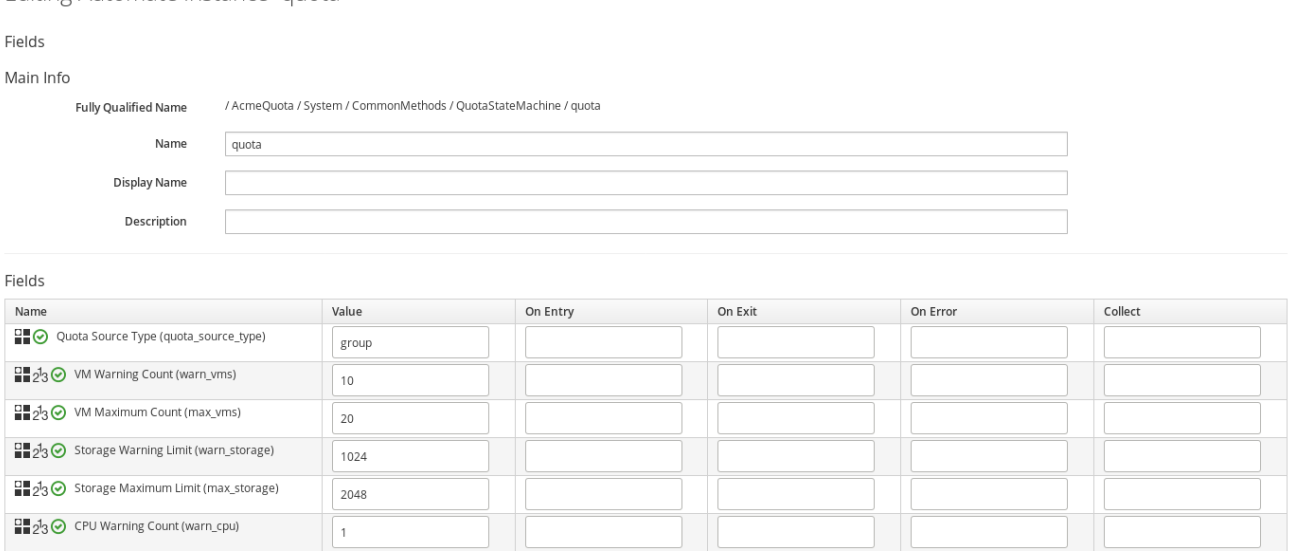
1. Log in as a user with administrator or super administrator rights to the Infrastructure Automation console.
2. Navigate to **Automation > Embedded Automate > Explorer**.
3. Copy the **ManageIQ > System > CommonMethods > QuotaStateMachine

   quota** instance to a custom DOMAIN.
4. From the accordion menu, click **DOMAIN > System > CommonMethods

   QuotaStateMachine > quota**.

   Note:

   By default, quotas are applied to tenants and do not require any change in **Automation > Embedded Automate > Explorer**.
5. Click (**Configuration**), (**Edit this instance**).

   1. Set the value for **Quota Source Type** to *user* or *group*.

      Important:

      A user creating a provisioning request must have an email address saved in their profile, or provisioning may fail. See Creating a User in *General Configuration* for details on configuring users.
   2. Set the values for **VM Warning Count**, **VM Maximum Count**, **Storage Warning Limit**, **Storage Maximum Limit**, **CPU Warning Count**, **CPU Maximum Count**, **Memory Warning Limit**, or **Memory Maximum Limit** to be the maximums for a specific user or group.
6. Click **Save**.

Using Tags for Owner and Group Quotas

Infrastructure Automation provides tags for enforcing quotas for the owners of virtual machines or instances. Ownership of a virtual machine or instance can be set either during the provisioning process or by using the **Configuration Set Ownership** button. If a virtual machine or instance has an owner, the value is displayed in the **Lifecycle** section of the virtual machine or instance summary page.

Quota tags can be assigned directly to **either** a group or owner **not** to a configuration item. The table below shows the tags for use in quotas.

Category Display Name (Name)	Use
Quota Max CPU (quota_max_cpu)	Sets the maximum number of CPUs summed over all virtual machines and instances owned by the group or user. Sample Values: 1, 2, 3, 4, 5, 10, 20, 30, 40, 50

Category Display Name (Name)	Use
Quota Max Memory (quota_max_memory)	Sets the maximum memory summed over all virtual machines and instances owned by the group or user. Sample Values: 1024, 2048, 4096, 8192, 10240, 20480, 40960, 81920 (in MB)
Quota Max Storage (quota_max_storage)	Sets the maximum storage summed over virtual machines and instances owned by the group or user. Sample Values: 10, 100, 1000, 20, 200, 40, 400 (in GB)

Applying a Tag to a User or User Group

1. Click (**Settings**) > **Application Settings**.
2. Click the **Access Control** accordion, and select the user or group that you want to tag.
3. Click (**Policy**), then click (**Edit Tags**).
4. Select the appropriate customer tag to assign, then the value.
5. Click **Save**.

Note:

When quotas are applied by both automate instance and tagging, the tagged values will have higher precedence.

State Machines

The automate state machine processes an ordered list of states. It can ensure the successful completion of a step before the next step is run, permit steps to be retried, allow setting a maximum time to retry the state before exiting, and number of retries before exiting the state. Before each state is executed, the **`On_Entry`** method is executed and after the state ends the **`On_Exit`** or **`On_Error`** method is executed based on how the state ends.

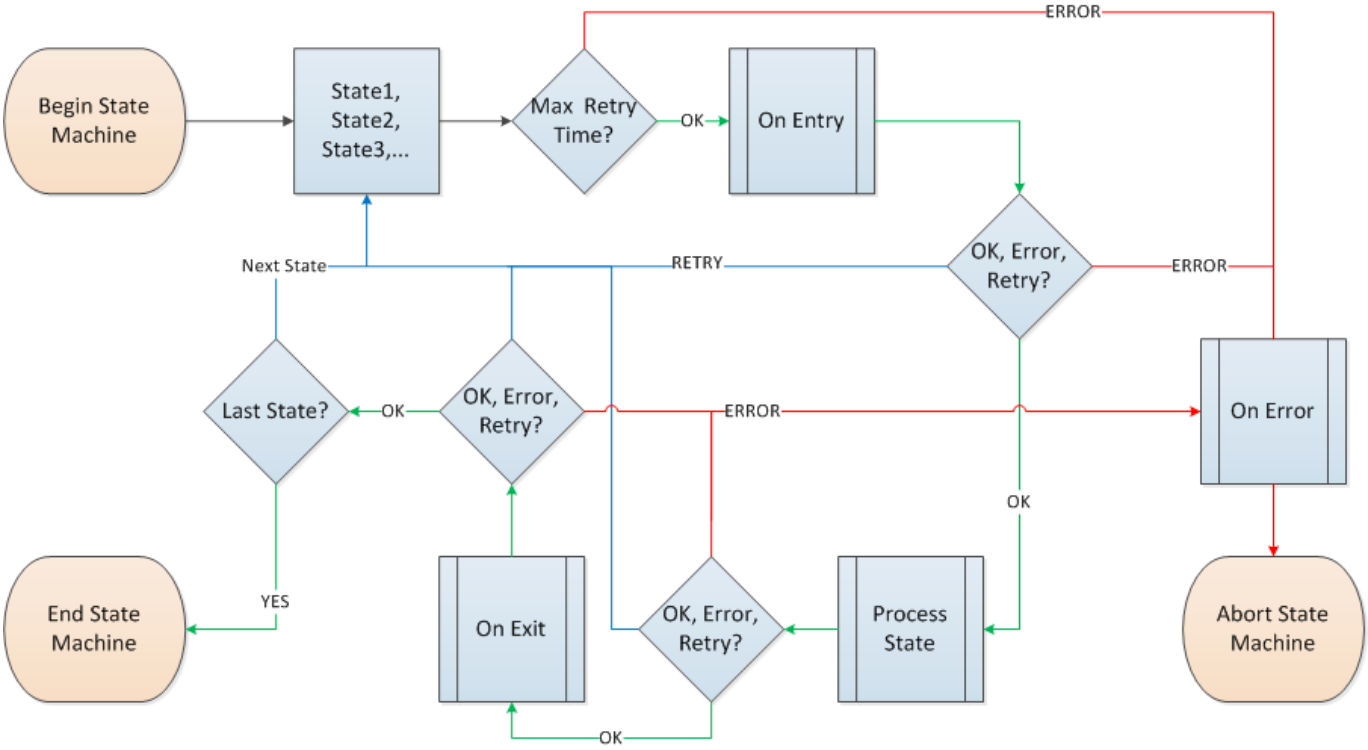
The following components make up a Infrastructure Automation automate state machine:

Component	Description
On_Entry	Method to run when entering the state. It enables you to execute an automate method to do some pre-processing before the state of the state machine is processed.
On_Exit	Method to run when exiting the state.
On_Error	Method to run if an error is encountered when running the state. It enables you to execute an automate method to do some final processing before the state machine finally exits (MIQ_ABORT) due to the error.
Default Value	Runs after the On_Entry method completes (The actual state being processed).
Max Retries	Maximum number of times to retry the state before exiting.
Max Time	Maximum time in seconds to retry the state before exiting.

In the diagram below, you can see how these components combine to create a state machine workflow.

Note:

The retry logic, **`On_Entry`** and **`On_Error`** are distinct cases in the program flow.



Code snippet demonstrating the state machine retry logic:

```
# Get current provisioning status
task = $evm.root['service_template_provision_task']
task_status = task['status']
result = task.status

Then check the result to see how it should proceed:

case result
when 'error'
  $evm.root['ae_result'] = 'error'
.....
when 'retry'
  $evm.root['ae_result'] = 'retry'
  $evm.root['ae_retry_interval'] = '1.minute'
when 'ok'
  $evm.root['ae_result'] = 'ok'
end

When the result is "retry", it sets:
  $evm.root['ae_result'] = 'retry'
  $evm.root['ae_retry_interval'] = '1.minute'
```

The following image shows a simple state machine pertaining to approving a provision request. This instance can be found in **Datastore > ManageIQ > Infrastructure > VM > Provisioning > StateMachines > ProvisioningRequestApproval > Default**.

Name	Value	On Entry	On Exit	On Error	Collect	Max Retries	Max Time	Message
max_vms	1							create
ValidateRequest		validate_request		pending_request		100		create
ApproveRequest		approve_request				100		create

1. The attribute **max_vms** has a value of 1. State machine processing can use the attributes of the state machine instance to make logic decisions. In this case, the **validate_request** method, which is processed during the **On_Entry** portion of the **ValidateRequest** state, evaluates the **max_vms** attribute. If the number of virtual machines requested is less than the **max_vms** value, the request can be auto-approved.
2. **ValidateRequest** is the first state to be executed.
3. **ApproveRequest** is the next state to be executed.

Note:

Grayed out items reflect values that are set in the class schema. These values can be overwritten on a per instance basis.

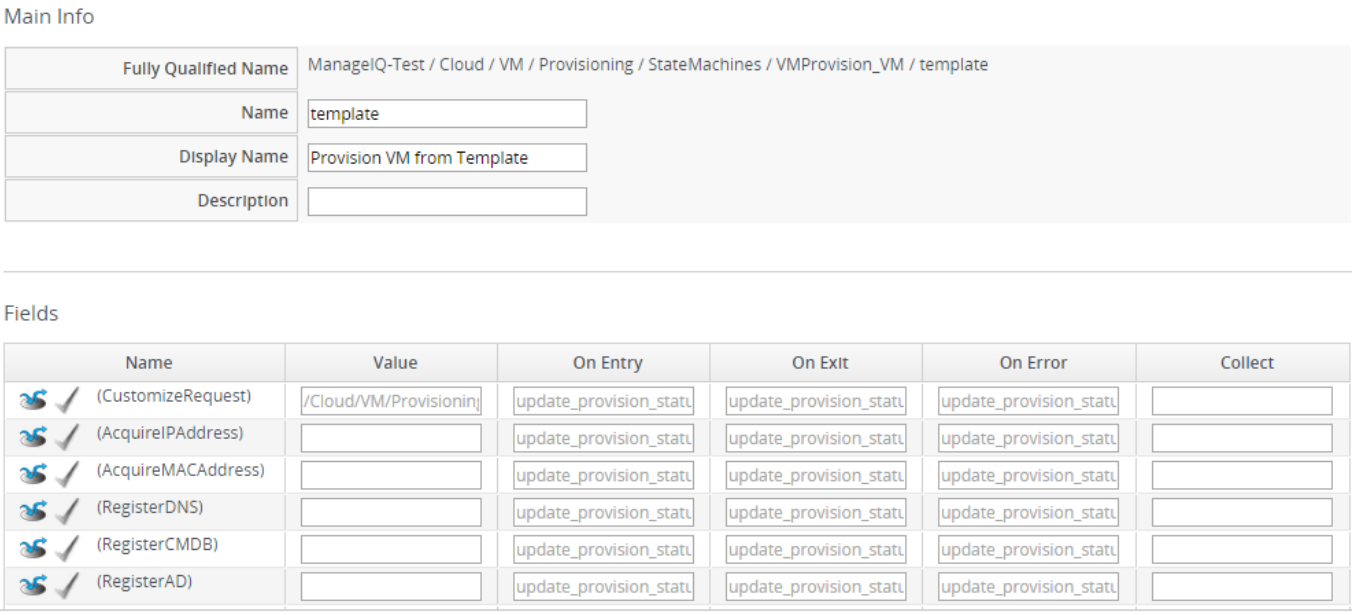
Customizing Provisioning States

The steps followed when provisioning a virtual machine or cloud instance are completed based on instances from the **DOMAIN > Cloud|Infrastructure > VM > Provisioning > StateMachines > VMProvision_VM** class.

Editing the Default State Instance

1. Navigate to **Automation > Embedded Automate > Explorer**.
2. Click **Datastore** accordion. Select **Configuration** > Add a New Domain**. You must fill the Name and Description values and click **Add** to create a custom domain.
3. From the accordion menu, navigate to **Cloud > VM > Provisioning > StateMachines > VMProvision_VM**.
4. Click on the drop-down menu, **Configuration > Copy this class**.
5. To add the class to the custom domain, specify the custom domain in the **To Domain** list and click Copy.
6. From the Datastore accordion, navigate to **custom domain > Cloud > VM > Provisioning > StateMachines > VMProvision_VM**.
    1. Click **Schema** tab.
    2. Select **Configuration > Edit selected Schema**.
7. You can remove, update, or insert a line in the Editing Class Schema panel.

Note: Depending on your environment, you can leave the values blank in a provisioning process. For example, if you are not integrating with IPAM or a CMDB, then you can leave the values blank.



1. On the Datastore tree, navigate to **custom domain > Cloud > VM > Provisioning > StateMachines > VMProvision_VM**. Click **Schema** tab and select Configuration > Edit sequence.
2. Resequence the schema in the Edit of Class Schema Sequence panel.

Viewing the Status of a Provisioning Request

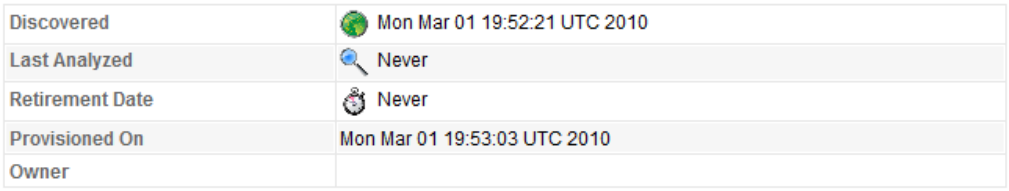
After a request has been approved, the various stages of fulfillment are executed. You can see the progress of the provisioning process by viewing its status.

1. Navigate to **Services > Requests**. The list of requests is shown.
2. Click on a specific request for more information. Once the provisioning begins, if the request was supposed to create more than one virtual machine or instance, a field will appear called **Provisioned VMs**. Click on the number that appears next to it for information on each of the individual provisions.

Viewing a Provisioned Virtual Machine or Instance

When a virtual machine or instance is created as a result of a provisioning request, its summary screen will show when it was provisioned in the **Lifecycle** area of the respective summary.

1. From **Services > Workloads**, click the virtual machine or instance that you want to view.



Viewing a Virtual Machine or Instance Summary

From **Services > Workloads**, click the virtual machine or instance that you want to view.

You can now request the log table from **Services > Request** to display logs related to a particular request.

Note: The table appears only if there are any logs to display. This table is sorted by the time, severity, and message.

Catalogs and Services

Through the use of catalogs, Infrastructure Automation provides support for multi-tier service provisioning to deploy layered workloads across hybrid environments. You can create customized dialogs that will give consumers of the services the ability to input just a few parameters and provision the entire service. The following table lists the terminology associated with catalogs that you will use within the Infrastructure Automation user interface for service provisioning.

Type	Information
Catalog Bundle	A group of templates.
Catalog Item	A single template.
Template	A template is a copy of a preconfigured virtual machine, designed to capture the installed software and software configurations, as well as the hardware configuration of the original virtual machine.
Dialog Tabs	Part of a service dialog.
Element	An item on a tab in a dialog. It can be a button, check box, drop down list, radio button, tag control, text area box, or a text box.
Provisioning Dialogs	Dialogs created for host provisioning, virtual machine migration, or virtual machine provisioning. The dialog name must be added to the appropriate provision instance to be processed.
Service Catalog	A catalog item or catalog bundle that is available for provisioning.
Service Dialogs	Made up of fully customizable tabs, items, and values for use with service provisioning.

Terminology

Generic Objects

Generic Objects are object-like entities, defined at runtime, that have unique names and user-defined attributes and relationships. Residing in the Automate Engine datastore, generic objects are designed to manage objects other than those related to private infrastructure, and public or private cloud providers.

Using automate requests, services and catalog items, generic objects can be directly accessed or passed, during any step, as a parameter to a service state machine. As a result, generic objects can be used to quickly add the capability to provision and collect data on resources not supported by Infrastructure Automation.

Viewing Generic Objects Classes

View a list of generic objects and click through to see detailed summary information for each object.

1. Navigate to **Automate > Generic Objects**.
2. Click on a generic object class in the table to view its summary information.

Creating Generic Objects Classes

Model a new resource by creating a generic object class and adding it to your Infrastructure Automation inventory. Each generic object class can have attributes, associations, and methods. Once created, generic object classes are visible to users of the Self Service user interface at the resource level.

Create a generic object class using the following steps:

1. Browse to menu: **Automation > Automate > Generic Objects**.
2. Click **Configuration**, then click (**Add a New Generic Object Class**).
3. Provide a **Name** and **Description** for the new object class.

4. In the **Attributes** field, enter a **Name** and choose a **Type** from the drop-down list. Click the + button to add attributes.
5. Enter a **Name** and select a **Class** for the object class's **Associations**. Click the + button to create additional associations.
6. Provide a **Name** for the **Methods**. Click the + button to add methods.
7. Click **Add**.

Editing Generic Object Classes

Edit existing generic object classes using the following steps:

1. Browse to menu: **Automation > Automate > Generic Objects**.
2. Click on a generic object class in the list view.
3. Click **Configuration**, then click (**Edit this Generic Object Class**).
4. Make required changes to the generic object class fields.
5. Click **Save**.

Removing Generic Objects Classes

Remove generic object classes from your inventory using the following steps:

1. Browse to menu: **Automation > Automate > Generic Objects**.
2. Check the generic objects classes from the table to remove.
3. Click **Configuration**, then click (**Remove selected Generic Object Classes from Inventory**).
4. Click **OK** to confirm.

Exporting Generic Objects

Export generic objects from Infrastructure Automation to create a shared library.

Exporting a generic object requires the object has been created in Infrastructure Automation.

Procedure

1. ssh to the Infrastructure Automation appliance as **root**.
2. Browse to **/var/www/miq/vmdb**:

   ```
   # cd /var/www/miq/vmdb
   ```

3. Create a temporary directory to store the generic object definitions:

   ```
   # mkdir tmp/generic_object_definitions
   ```

4. Export the generic object definitions using the following **bin/rake** command:

   ```
   # bin/rake evm:export:generic_object_definitions -- --directory tmp/generic_object_definitions
   ```

Locate the exported yaml file containing the generic object properties in the temporary folder.

Importing Generic Objects

Import generic objects to Infrastructure Automation from a shared library.

Procedure

1. ssh to the Infrastructure Automation appliance as **root**.
2. Browse to **/var/www/miq/vmdb**:

   ```
   # cd /var/www/miq/vmdb
   ```

3. Create a temporary directory to store the generic object definitions:

   ```
   # mkdir tmp/generic_object_definitions
   ```

4. Copy the yaml file containing the generic object definitions to the temporary folder:

```
# cp generic_objects.yaml tmp/generic_object_definitions
```

5. Import the generic object definitions to the Infrastructure Automation appliance using the following **bin/rake** command:

```
# bin/rake evm:import:generic_object_definitions -- --source tmp/generic_object_definitions
```

Confirm the generic objects have imported successfully by Viewing Generic Objects Classes in the Infrastructure Automation user interface.

Service Dialogs

When provisioning a service, input will be needed from the requester. Service dialogs are used to take input from the user. This input is connected to a method in the Automate model that defines how the user's input is translated into the provision request. Before creating a service dialog, be sure to plan what items you need the user to input.

Adding a Service Dialog

- When creating a service dialog for use with Ansible playbook catalog items, variable elements must use the prefix **param_** when assigning the value. For example, a new variable labeled **key1** should have its value set as **param_key1**. - Using Ansible playbooks to populate dynamic dialog fields is not recommended due to delay times caused by the overhead of interaction between systems. - If you add the playbook automate method to a service dialog, only users with admin privileges can run the dialog.
Infrastructure Automation includes a drag-and-drop service dialog editor to create service dialogs. The editor, with its drag-and-drop feature, provides a visual representation of the components that comprise a service dialog. You can easily design your service dialog utilizing dialog tabs, sections (previously referred to as boxes), and elements.

When users access a service, the majority of options available to them are preset and cannot be altered. The requirements for the service determine the options and fields that need to be present in the dialog for user input. A service dialog exposes some of those options to the user so that even if they are ordering a basic Red Hat Enterprise Linux 7 machine, for example, they can at least choose the amount of memory, virtual CPUs, or other options available to the instance they order. In cases where certain fields must be unique, such as the name of virtual machines in Red Hat Virtualization, users must enter their own unique name for the virtual machine they choose or the operation will fail, so this field must be exposed.

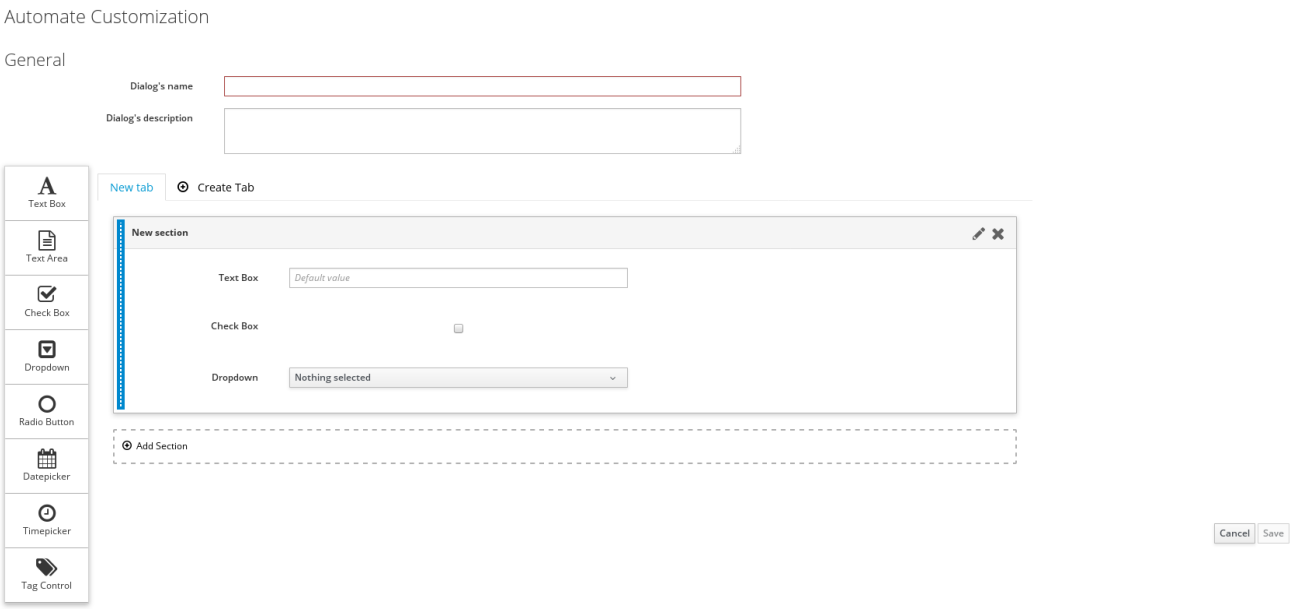
A service dialog contains three components:

- One or more **Tabs**.
- Inside the tabs, one or more **Sections**. Note that in the previous method of creating service dialogs using the Infrastructure Automation user interface, **Sections** were referred to as **Boxes**.
- Inside the sections, one or more **Elements**. Elements are controls that accept input. Elements contain methods, like check boxes, drop-down lists or text fields, to fill in the options on the provisioning dialog.

Important:

The names of the elements must correspond to the options used in the provisioning dialog.

1. Browse to menu: **Automation > Automate > Customization**.
2. Click the **Service Dialogs** accordion.
3. Click **Configuration**, and then **Add a new Dialog**.



4. Enter basic details under **General**:
    1. Enter the **Dialog's name** and **Dialog's description**.
5. Add a new tab to the dialog:
    1. Click **Create Tab**. Then, click the icon on the new tab to edit tab information.
    2. Enter a **Label**.
    3. Optional: Enter a description for the tab in **Description**.
    4. Click **Save**.
6. Add a new section to the tab:
    1. Click **Add Section**. Then, click the icon on the upper-right to edit section details.
    2. Enter a **Label**.
    3. Optional: Enter a description for the section in **Description**.
    4. Click **Save**.
7. Add elements to the section:
    1. From the list of elements on the left, click an element you want to add, then drag-and-drop it inside the section. Then, click the icon next to the element to edit its field details.

Element Types	Additional Info
Text Area	Provides text area for users to enter text. You can enter the default text in **Default Value**, or leave it as blank.
Text Box	Text box is similar to a text area with the added option to enable **Protected** so the text is shown as asterisks(*), instead of plain text.
Check Box	Enable **Default Value** if you want the box checked by default.
Drop Down	Use drop down to create list entries either manually or using automate methods. Enable **Dynamic** to create lists using automate methods; use **Entry Point** to select an automate instance. Enable **Show Refresh Button** to allow users to refresh list options manually.
Radio Button	Similar to a drop down but displays options using radio buttons.
Datepicker	Use this to enable users to pick a date by clicking the calendar icon.
Timepicker	use this to enable users to pick a date and time.
Tag Control	Select a **Category** of tags you want assigned to virtual machines associated with the service dialog. Enable **Single Select** if only one tag can be selected.

    2. Enter a **Label**, **Name**, and **Description** for the element.

       Element names must correspond to the options used in the provisioning dialog. **Name** must use only alphanumeric characters and underscores without spaces. It is also used to retrieve the value of this element in the method used with the dialog and must start with **dialog_service_type**.

3. Optional: Add additional information in **Help** to assist the user to complete the fields in the service dialog. This field is useful for explaining unfamiliar terminology or providing configuration tips. This information is presented when you hover over the [!] exclamation mark in the Service Dialog while ordering a Service Catalog later.
4. Set other options as required.
5. Click **Save**.

8. Optional: Repeat the above step to add more elements to the existing section, or create and add elements to a new section as required.
9. Optional: Repeat the step to add a new tab to the dialog, and subsequent steps to add sections and elements to it as required.
10. Click **Save** to create the dialog.

The service dialog is now created, and added to the **Service Dialogs** accordion.

Creating a Service Dialog from a Container Template

Complete the following procedure to create a Service Dialog from a Container Template.

1. Navigate to **Compute > Containers > Container Templates** and select the template for provisioning.
2. Click (**Configuration**), then (**Create Service Dialog from Container Template**).
3. Enter a name for the dialog in **Service Dialog Name**.
4. Click **Save**.

You can use this service dialog when creating a catalog item for container template provisioning; see Creating an OpenShift Template Catalog Item.

Importing Service Dialogs

You can share service dialogs between appliances using the export and import features.

1. Navigate to **Automation > Embedded Automate > Customization**.
2. In the **Import/Export** accordion, click **Service Dialog Import/Export**.
3. In the **Import** area, click **Browse** to select an import file.
4. Click **Upload**.

Exporting Service Dialogs

You can share service dialogs between appliances using the export and import features.

1. Navigate to **Automation > Embedded Automate > Customization**.
2. In the **Import/Export** accordion, click **Service Dialog Import/Export**.
3. In the **Export** area, select the service dialogs that you want to export.
4. Click **Export**.

Methods

Creating a Method to Associate with the Dialog

You will need to create a method that connects the values in the dialog with the provisioning request. The method should be created in the `DOMAIN/Service/Provisioning/StateMachines/ServiceProvision_Template` class of the **Automate** model.

Note:

DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.

A method is provided below that was created for the following scenario:

- You want to provision a three-tiered service that contains catalog items of web, app and DB. Each of these virtual machines (or cloud instances) has been tagged under the **Service** category with the appropriate value. Then, added as a catalog item and combined into a catalog bundle.

- The **Service Dialog** captures the selection of small, medium or large application in a dropdown called **service_type**. When referring to a value captured in an element in a dialog, the name of the element should be prefixed with **dialog_**. For example, **service_type** becomes **dialog_service_type** when used in the method.
- The method will set the memory sizes for each of the catalog items based on the **service_type** selection.

```
#            Automate Method
#
$evm.log("info", "Automate Method ConfigureChildDialog Started")
#
#            Method Code Goes here
#
$evm.log("info", "==========================================")
$evm.log("info", "Listing ROOT Attributes:")
$evm.root.attributes.sort.each { |k, v| $evm.log("info", "\t#{k}: #{v}")}
$evm.log("info", "==========================================")

stp_task = $evm.root["service_template_provision_task"]
$evm.log("info", "==========================================")
$evm.log("info", "Listing task Attributes:")
stp_task.attributes.sort.each { |k, v| $evm.log("info", "\t#{k}: #{v}")}
$evm.log("info", "==========================================")

#############################################################
#### This is how the method would look for dialog variables
#############################################################
dialog_service_type = $evm.root['dialog_service_type']
$evm.log("info","User selected Dialog option = [#{dialog_service_type}]")

stp_miq_request_task = stp_task.miq_request_task
#$evm.log("info","(parent) miq_request_task:  = [#{stp_miq_request_task}]")

#############################################################
#### This is how you get the catalog items for the catalog bundle
#############################################################

stp_miq_request_tasks = stp_task.miq_request_tasks
#$evm.log("info","(children) miq_request_tasks count:  = [#{stp_miq_request_tasks.count}]")

#############################################################
#### By going through the children, you can set the dialog variable for each of the children (we based our
values on the childrens service tags)
#############################################################

stp_miq_request_tasks.each do |t|

  $evm.log("info"," Setting dialog for: #{t.description}")
  service = t.source
  service_resource = t.service_resource
  #$evm.log("info"," Child service resource name: #{service_resource.resource_name}")
  #$evm.log("info"," Child service resource description: #{service_resource.resource_description}")

  service_tag_array = service.tags(:app_tier)
  service_tag = service_tag_array.first.to_s

  memory_size = nil

#############################################################
#### The dialog_service_type is the attribute set on the service dialog
#### We use the service_tag to decide what child gets what dialog
#############################################################

  case dialog_service_type
  when "Small"
    case service_tag
    when "app"
      memory_size = 1024
    when "web"
      memory_size = 1024
    when "db"
      memory_size = 4096
    else
      $evm.log("info","Unknown Dialog type")
    end
  when "Large"
    case service_tag
    when "app"
      memory_size = 4096
    when "web"
      memory_size = 4096
    when "db"
```

```
      memory_size = 8192
    else
      $evm.log("info","Unknown Dialog type")
    end
  else
    $evm.log("info","Unknown Dialog type - setting Dialog options here")
  end

#############################################################
#### set_dialog_option sets the dialog for the child
#############################################################

  t.set_dialog_option('memory',memory_size) unless memory_size.nil?
  $evm.log("info","Set dialog for selection: [#{dialog_service_type}]  Service_Tier: [#{service_tag}] Memory
size: [#{memory_size}]")

end
#
#
#
$evm.log("info", "Automate Method ConfigureChildDialog Ended")
exit MIQ_OK
```

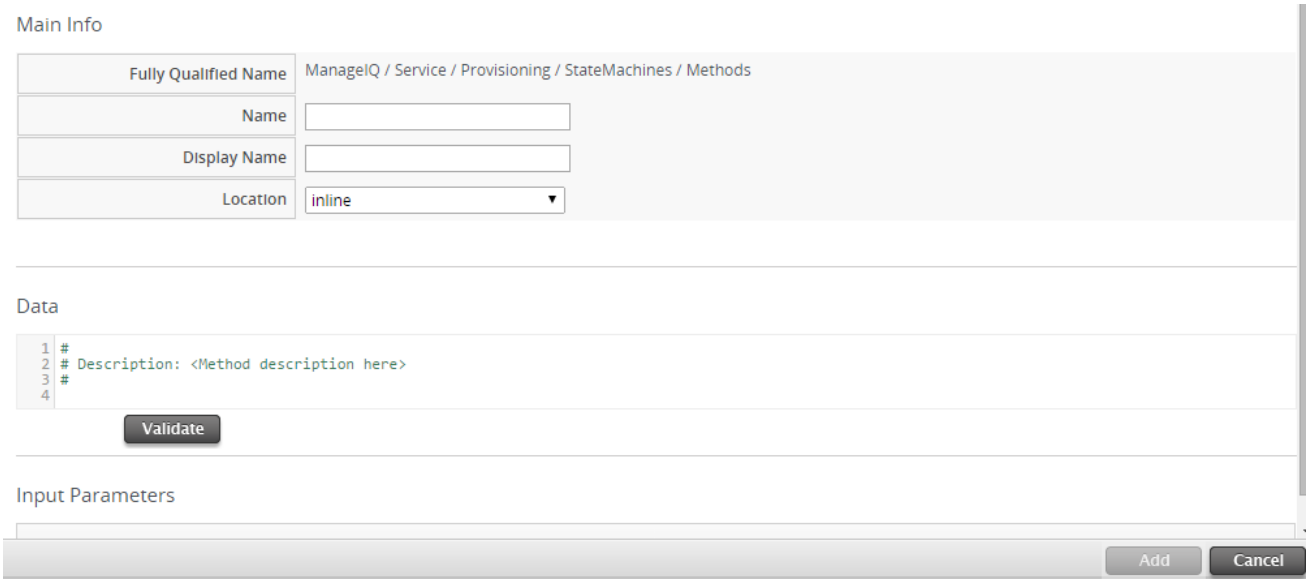
Creating a Method in the Service Class

Service methods have been split based on purpose.

1. Navigate to **Automation > Embedded Automate > Explorer**.
2. Service Class is located at **DOMAIN > Service > Provisioning > StateMachines > Methods** and **DOMAIN > Service > Retirement > StateMachines > Methods**.

   Note:

   DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.
3. Click the **Methods** tab.
4. Click (**Configuration**), then (**Add a New Method**).
5. Enter a **Name** and **Display Name**.
6. In the **Data** field, enter the method contents.
7. Click **Validate** and wait for your data entry to be successfully validated.
8. Click **Add**.



Creating an Instance in the Service Class

1. Navigate to **Automation > Embedded Automate > Explorer**.

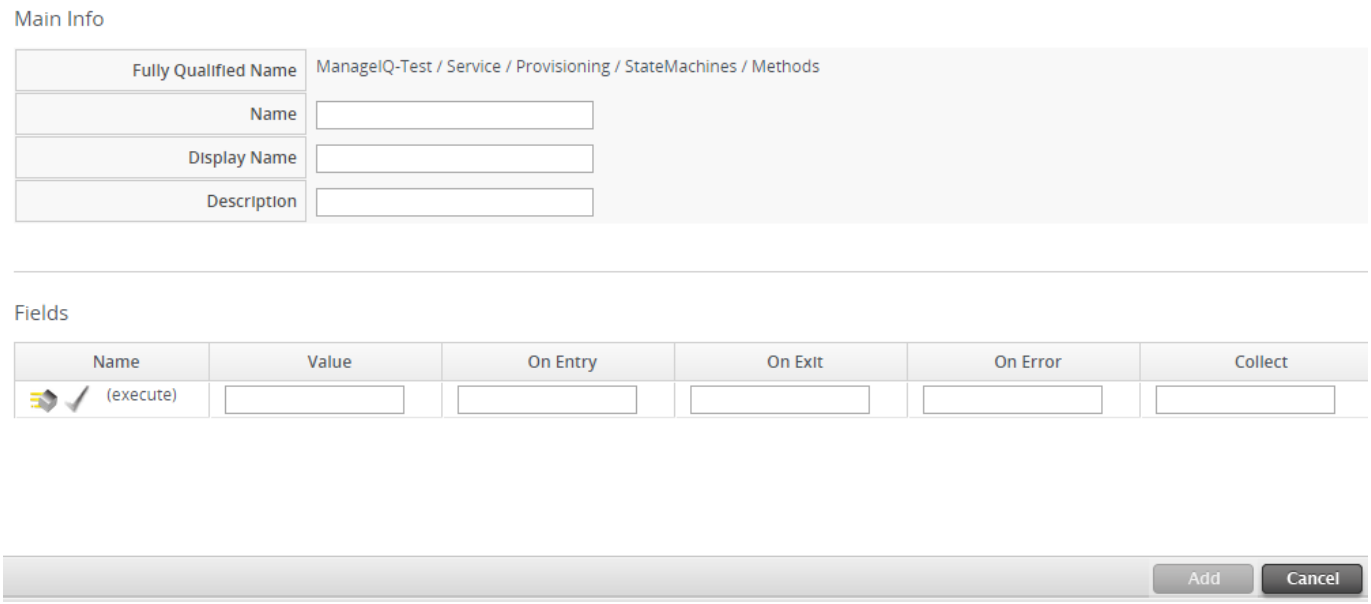
2. Service Class is located at **DOMAIN > Service > Provisioning > StateMachines > Methods** and **DOMAIN > Service > Retirement > StateMachines > Methods**.

   Note:

   DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.

3. Click the **Instances** tab.
4. Click (**Configuration**), then (**Add a new Instance**).
5. Enter a **Name** and **Display Name**.
6. In the **Fields** area, enter the method's name in **Value**.
7. Click **Add**.

The instance is created so that it can be called from the **ServiceProvision** class.



Note:

After the method has been created, it must be mapped to an instance in the `DOMAIN/Service/Service/Provisioning/StateMachines` class. The name of the instance must be specified as the **Entry Point**. This method must be called before the provision job begins.

Associating a Method with an Automate Instance

Service methods have been split based on purpose.

1. Navigate to **Automation > Embedded Automate > Explorer**.
2. From the accordion menu, click the required service method.
3. Service Class is located at **DOMAIN > Service > Provisioning > StateMachines > Methods** and **DOMAIN > Service > Retirement > StateMachines > Methods**.

   Note:

   DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.

4. Either create a new instance or select the **clone_to_service** instance.
5. Click (**Configuration**), then (**Edit Selected Instance**).
6. In the **configurechilddialog** value, put the path to the method.
7. Click **Save** or **Add** if you are adding this to a new instance.

Catalogs

Catalogs are used to create groups of virtual machines or instances for provisioning. For example, a complete package of a database server, desktop with specialized software already on it, and a firewall. You will need to complete the following steps to create and provision a service catalog.

1. Create **Catalog Items** for each virtual machine or instance that will be part of the service.
2. Create a **Service** dialog. For example, create a dropdown with three options small, medium, and large.
3. Create a method for the Service Dialog. This method defines what each of the options means to each of the individual virtual machines or cloud instances for the service. This method is called from a service provisioning instance in the Automate model.
4. Create an instance in the **`DOMAIN/Service/Provisioning/StateMachines/ServiceProvision_Template`** class that calls the method.

   Note:

   DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.
5. Associate method with Automate instance.
6. Create a **Catalog Bundle**, adding each of the catalog items to it. Select the **Service Dialog** you created. Use the instance created in the **`DOMAIN/Service/Provisioning/StateMachines/ServiceProvision_Template`** class as the **Entry Point**. Check **Display in Catalog** box.
7. Provision a service.

Creating a Catalog Bundle

1. Navigate to **Services > Catalogs**.
2. Click the **Catalog Items** accordion.
3. Click (**Configuration**), and then (**Add a New Catalog Bundle**).
4. Enter a name and description for the bundle. Select **Display in Catalog**.
5. Select the required **Catalog**.
6. Select the required **Dialog**.
7. Select chargeback currency.
8. Enter **Price / Month**.
9. Select the provisioning and retirement entry points.
10. Select **Additional Tenants**.
11. Click on the **Resources** tab, then select the catalog item you want to add to the bundle from the **Add a Resource** dropdown.
12. Click **Add**.

A catalog bundle is created and visible in the **Service Catalog** accordion.

Note:

You should also create and specify an Entry Point in the **`DOMAIN/Service/Provisioning/StateMachines/Methods/CatalogBundle`** class for each catalog item that is part of a bundle. If you do not, then the pre and post provision processing will occur for each item in the bundle in addition to processing for the **Catalog Bundle**. To set the entry point, go into each **Catalog Item** and check **Display in Catalog**. Then, you will see the **Entry Point** field.

Copying a Catalog Bundle

1. Navigate to **Services > Catalogs**.
2. Click the **Catalog Items** accordion.
3. Select the catalog bundle you want to copy from **All Catalog Items**.
4. Click (**Configuration**), and then (**Copy Selected Item**).
5. Enter a name for the copy of catalog bundle you are creating. Note that this name must not already be in use.

6. Click **Add**.

A copy of the catalog bundle is saved. You can now select this new copy of the catalog bundle and edit as required by navigating to (**Configuration**), then clicking (**Edit this Item**).

Creating a Catalog Item

Create a catalog item for each virtual machine or cloud instance that will be part of the service.

1. Browse to menu: **Services > Catalogs**.
2. Click the **Catalog Items** accordion.
3. Click **Configuration**, and then (**Add a New Catalog Item**).
4. Select the **Catalog Item Type** you are adding. This list only shows items related to providers available or options activated in the appliance (for example, **Ansible Playbook** is available as a **Catalog Item Type** option if the Embedded Ansible server role is enabled on the appliance).
5. Enter a name and description for the bundle. Select **Display in Catalog**.
6. Select a **Catalog**.
7. Select a **Dialog**.

   Note:

   You can only choose from the catalogs and dialogs you have already created. If you haven't done so, leave the values blank and edit later.
8. Select a **Zone**.
9. Select the chargeback currency.
10. Enter **Price / Month**.
11. Select the catalog item's **Subtype**.
12. Select the provisioning and retirement entry points, that is the Automate instance to run upon provisioning and retirement.

    Note:

    The entry point must be a State Machine since the **Provisioning Entry Point** list is filtered to only show State Machine class instances. No other entry points will be available from the **Provisioning Entry Point** field.
13. Select **Additional Tenants**.
14. Click **Add**.

Copying a Catalog Item

Note:

When copying a catalog item for reuse, you must click **Display in Catalog** in the copied catalog item for the item to appear in the catalog.

1. Navigate to **Services > Catalogs**.
2. Click the **Catalog Items** accordion.
3. Select the catalog item you want to copy from **All Catalog Items**.
4. Click (**Configuration**), and then (**Copy Selected Item**).
5. Enter a name for the copy of catalog item you are creating. Note that this name must not already be in use.
6. Click **Add**.

A copy of the catalog item is saved. You can now select this new copy of the catalog item and edit as required by navigating to (**Configuration**), then clicking (**Edit this Item**).

Creating a Generic Catalog Item

Create generic catalog items for services non-specific to virtualization or cloud environments. This catalog item type can serve a wide array of needs, from creating a vLAN across a network to accessing virtual machine IP addresses and adding them to a load balancer pool.

1. Browse to menu: **Service > Catalogs**.
2. Click the **Catalog Items** accordion.
3. Click **Configuration**, and then (**Add a New Catalog Item**).
4. Select **Generic** from the Catalog Item Type list.
5. In the **Basic Info** subtab:
    1. Type a **Name/Description**.
    2. Check **Display in Catalog** to display the item in the catalog. A **Dialog** will be required if you select **Display in Catalog**.
    3. Choose a **Catalog** to which to add the new item.
    4. Select a **Dialog** from the available options.
    5. Choose a **Subtype** from the list menu.
    6. Add **Entry Point(NS/Cls/Inst)** options.
        1. **Provisioning Entry Point (Domain/NS/Cls/Inst)** requires you to select an Automate instance to run upon provisioning.
        2. **Retirement Entry Point (Domain/NS/Cls/Inst)** requires you to select an Automate instance to run upon retirement.

            Note:

            The entry point must be a State Machine since the **Provisioning Entry Point** list is filtered to only show State Machine class instances. No other entry points will be available from the **Provisioning Entry Point** field.
6. In the **Details** subtab, write a **Long Description** for the catalog item.
7. Click **Add**.

Creating an Ansible Playbook Service Catalog Item

Create a catalog item that uses an Ansible Playbook to back it.

Note:

- Before creating an Ansible service, at least one repository, one playbook, and one credential must exist in the Infrastructure Automation inventory. Check your inventory and add the appropriate resources before creating an Ansible service. For more information, see Automation Management Providers in *Managing Providers*.
- Debugging verbosity is available for Ansible playbook catalog items. Selecting a higher verbosity value provides more detailed output as the playbook executes. **0 (Normal)** is the default value. **1 (Verbose)** will yield return data while a value of **3 (Debug)** provides connection attempt and task invocation details. Higher levels, such as **4 (Connection)** can be useful for debugging SSH connections. Use **5 (WinRM Debug)** when debugging WinRM connections.
- Using Ansible playbooks to populate dynamic dialog fields is not recommended due to delay times caused by the overhead of interaction between systems.
- Only users with administrator privileges can run a service dialog based on a playbook.

1. Browse to menu: **Services < Catalogs**.
2. In the **Catalog Items** accordion, click on the **All Catalog Items**.
3. Click **Configuration**, then (**Add a New Catalog Item**).
4. Select **Ansible Playbook** from the **Catalog Item Type** drop-down list.
5. Type a **Name** and **Description** for the new service catalog item.
6. Click **Display in Catalog**.
7. Select the appropriate **Catalog** from the drop-down list.
8. In the **Provisioning** tab, set parameters for your catalog item to use by configuring a Playbook to back your service item:
    1. Choose a **Repository** from the drop-down list.
    2. Select the **Ansible Playbook** to use.

3. Assign the appropriate **Machine Credentials** from the drop-down list.
4. Add **Cloud** or **Network Credentials** from the drop-down lists.
5. Choose the **Host** against which to run the service item.
6. Set the **Max TTL** in minutes. The Time To Live (TTL) field allows you to set the maximum execution time for the playbook to run.
7. Use the **Escalate Privilege** toggle switch to enable user privilege escalation if called for in credentials during the playbook run.
8. Choose a **Verbosity** value to set the debug level for playbook execution.
9. Add key value pairs for **Variables** and their corresponding **Default Values**.
10. In the **Dialog** options, choose an existing dialog from the **Use Existing** drop-down list or select **Create New** to add a new dialog.

9. In the **Retirement** tab, set parameters for your catalog item to use by selecting values for the following:
    1. Choose a **Repository** from the drop-down list.
    2. Select the **Ansible Playbook** to use.
    3. Assign the appropriate **Machine Credentials** from the drop-down list.
    4. Add **Cloud** or **Network Credentials** from the drop-down lists.
    5. Choose the **Host** against which to run the service item.
    6. Set the **Max TTL** in minutes. The Time To Live (TTL) field allows you to set the maximum execution time for the playbook to run.
    7. Use the **Escalate Privilege** toggle switch to enable user privilege escalation if called for in credentials during the playbook run.
    8. Choose a **Verbosity** value to set the debug level for playbook execution.
    9. Add key value pairs for **Variables** and their corresponding **Default Values**.
    10. In the **Dialog** options, choose an existing dialog from the **Use Existing** drop-down list or select **Create New** to add a new dialog.
10. Click **Add**.

Creating an Ansible Tower Service Catalog Item

Create a service catalog item from an Ansible Tower template you can use to execute an Ansible Tower playbook in Infrastructure Automation.

You must first create the job or workflow template in Ansible Tower. The job or workflow templates are automatically discovered by Infrastructure Automation when refreshing your Ansible Tower provider's inventory.
First, create a catalog:

1. Browse to menu: **Services > Catalogs**.
2. Click **Configuration**, then (**Add a New Catalog**)
3. Enter a **Name** and **Description** for the catalog.
4. Click **Add**.

Then, create an Ansible Tower service catalog item:

1. Browse to menu: **Automation < Ansible Tower < Explorer**, then click on the **Templates** accordion menu.
2. Click **Ansible Tower Templates** and select an Ansible Tower job or workflow template.
3. Click **Configuration**, then (**Create Service Dialog from this Template**).
4. Enter a **Service Dialog Name** (for example, *ansible_tower_job*)and click **Save**.
5. Browse to menu: **Services > Catalogs**. Click **Catalog Items**.
6. Click **Configuration**, then (**Add a New Catalog Item**) to create a new catalog item with the following details, at minimum:
    - For **Catalog Item type**, select **Ansible Tower**.
    - Enter a **Name** for the service catalog item.
    - Select **Display in Catalog**.

- In **Catalog**, select the catalog you created previously.
- In **Dialog**, select the service dialog you created previously (in this example, *ansible_tower_job*). To ask the user to enter extra information when running the task, **Service Dialog** must be selected. A dialog is required if **Display in Catalog** is chosen.
- In **Provider**, select your Ansible Tower provider. This brings up the **Ansible Tower Template** option and configures the **Provisioning Entry Point State Machine** automatically.
- Add configuration information for **Reconfigure Entry Point** and **Retirement Entry Point** as applicable.
- Select your desired **Ansible Tower Template** from the list. Generally, this is the Ansible Tower job or workflow template previously used to create the service dialog.

7. Click **Add**. The catalog item you created will appear in the **All Service Catalog Items** list.

Creating an Amazon Service Catalog Item

Use the following procedure to create an Amazon catalog item. Once created, the catalog item and service dialog combine with all of the options in the provisioning dialog. Users can then order Red Hat Enterprise Linux instances from the **Service Catalog** in the Infrastructure Automation Service user interface.

1. Browse to menu: **Services > Catalogs**, then click on the **Catalog Items** accordion.
2. Click **Configuration**, then **Add a New Catalog Item**.
3. Select **Amazon** from the **Catalog Item Type** list.
4. Enter the basic details in the **Basic Info** tab:
    1. Enter a **Name** and **Description** for the new service catalog item.
    2. Select **Display in Catalog**.
    3. Select the appropriate catalog from the **Catalog** list.
    4. Select the appropriate service dialog from the **Dialog** list.
5. Click the **Request Info** tab to enter request details:
    1. On the **Catalog** tab, select your Amazon AWS image name from the **Name** list, and the number of instances from the **Count** list. The **VM Name** will be overwritten during the provisioning process, but you can enter it as *changeme* for now.
    2. On the **Properties** tab, select *T2 Micro* from the **Instance Type** list, and *Basic* or *Advanced* for **CloudWatch**. If you plan to access the instance, select a **Guest Access Key Pair**, too.
    3. On the **Customize** tab, set the **Root Password** under **Credentials**, then select the *Basic root pass template* as a script for cloud-init under **Customize Template**.
6. Click **Add**.

Creating an Azure Service Catalog Item

Use the following procedure to create an Azure catalog item.

1. Browse to menu: **Services > Catalogs**, then click on the **Catalog Items** accordion.
2. Click **Configuration**, then (**Add a New Catalog Item**).
3. Select **Azure** from the **Catalog Item Type** list.
4. Enter the basic details in the **Basic Info** tab:
    1. Enter a **Name** and **Description** for the new service catalog item.
    2. Select **Display in Catalog**.
    3. Select the appropriate catalog from the **Catalog** list.
    4. Select the appropriate service dialog from the **Dialog** list.
5. Click the **Request Info** tab to enter request details:
    1. On the **Catalog** tab, select your Azure image name from the **Name** list, and the number of instances from the **Count** list. The **VM Name** will be overwritten during the provisioning process, but you can enter it as *changeme* for now.

2. Select appropriate **Environment** settings that are known to work for your Azure environment.
3. On the **Customize** tab, set the **Username** and **Password** under **Credentials**, then select the appropriate script under **Customize Template**.

6. Click **Add**.

Creating an OpenShift Template Catalog Item

Note:

Before adding a new catalog item for container template provisioning, create a service dialog from a container template. See Creating a Service Dialog from Container Template for details.

Complete the following procedure to create an OpenShift Template catalog item.

1. Browse to menu: **Service > Catalogs**, then click on the **Catalog Items** accordion.
2. Click **Configuration**, then (**Add a New Catalog Item**).
3. Select **OpenShift Template** from the **Catalog Item Type** list.
4. Enter a **Name** and **Description** for the new service catalog item. Select **Display in Catalog**.
5. Select the appropriate catalog from the **Catalog** list.
6. From the **Dialog list**, select the service dialog you have created from a container template.
7. Select your provider from the **Provider** list.
8. Set the **Provisioning Entry Point**.
9. Click **Add**.

Creating an Orchestration Catalog Item

Use the following procedure to create an Orchestration catalog item.

1. Browse to menu: **Services > Catalogs** and select **Catalog Items** in the accordion menu.
2. Click **Configuration**, then click **Add a New Catalog Item**. The **Adding a new Service Catalog Item** window is displayed.
3. Select **Orchestration** from the **Catalog Item Type** list.
4. Enter the basic details in the **Basic Info**:
    1. Enter a **Name** and **Description** for the new service catalog item.
    2. Select **Display in Catalog** box.
    3. Select the appropriate catalog from the **Catalog** list.
    4. Select the appropriate dialog from the **Dialog** list.
    5. Select the **Orchestration Template** from the list.
5. Click **Add**.

Creating a VMware Content Library OVF Template Catalog Item

Use the following procedure to create a VMware Content Library OVF Template catalog item.

Note:

Before you create the catalog item, make sure you have added a VMware vCenter Provider, and create a Service Dialog. See Adding a VMware vCenter Provider and Creating a Service Dialog for more information.

1. Browse to menu: **Services > Catalogs** and select **Catalog Items** in the accordion menu.
2. Click **Configuration**, then click **Add a New Catalog Item**. The **Adding a new Service Catalog Item** window is displayed.
3. Select **VMware Content Library OVF Template** from the **Catalog Item Type** list.
4. Enter the values in the **Basic Info**:

1. Enter a **Name** and **Description** for the new service catalog item.
2. Select **Display in Catalog** box.
3. Select the appropriate catalog from the **Catalog** list.
4. Select the appropriate dialog from the **Dialog** list.
5. Select the **OVF Template** that you want to use for the deployment.
6. Enter a **VM Name**.
7. Select the **Accept EULA** box.
8. Select the **Resource Pool**.
9. Optional: Select the **Folder** and the **Host**.

5. Click **Add**.

Provisioning a Service

1. Browse to menu: **Services > Catalogs**.
2. Click the **Service Catalogs** accordion, and select the service to provision.
3. Click **Order**.
4. On the **Order Service** screen, select the box for **Do you want to continue?** and click **Submit**.

The parameters are passed to the children based on the method tied to the choices made in the dialog.

Orchestration Stacks

Cloud orchestration is a service that allows you to create, update, and manage cloud resources and their software components as a single unit and then deploy them in an automated, repeatable way through a template. Templates use a human-readable syntax and can be defined in text files, thereby allowing users to check them into version control. Templates allow you to easily deploy and reconfigure infrastructure for applications within your cloud. A user can author the stack templates, or can upload them from other sources.

Infrastructure Automation supports adding Amazon CloudFormation, OpenStack Heat, Microsoft Azure, VNF, and VMware vApp template type, and provides the ability to:

- Inventory stacks and elements of each type into the Infrastructure Automation VMDB.
- Model the relationships of instances to their stacks, inclusive of the user interface. For example, selecting an instance within a region that is within a stack, the user interface shows this on the standard instance view.
- Model the stack and its elements in the user interface.

Note:

When importing a template into Infrastructure Automation, the selected elements are converted according to their type. For example, lists convert to list boxes, and single items convert to text boxes.

Creating an Orchestration Template

Complete the following procedure to add an orchestration template.

1. Navigate to **Services > Catalogs** and select **Orchestration Templates** in the accordion menu.
2. Click **Configuration**, then click **Create a new Orchestration Template**.
3. Enter a **Name** and **Description** for your template.
4. Select the template type from the **Template Type** list. The default is Amazon CloudFormation.
5. Select **Draft** to create a draft template.
6. Add your template in the area below for the selected **Template Type**.
7. Click **Add**.

Editing Orchestration Templates

Complete the following procedure to edit orchestration templates.

1. Navigate to **Services > Catalogs** and select **Orchestration Templates** in the accordion menu.
2. Select the orchestration template you want to edit from the **All Orchestration Templates** list.
3. Click **Configuration**, then click **Edit this Orchestration Template**.
4. Edit the template as needed.

   Note:

   You can only edit the name and description of a read-only template as there can be stacks associated with the template.
5. Click **Save**.

Copying Orchestration Templates

Complete the following procedure to copy an orchestration template to create a new template.

1. Navigate to **Services > Catalogs** and select **Orchestration Templates** in the accordion menu.
2. Click **Configuration**, then click **Copy this Orchestration Template**.
3. Change the **Description** and the actual content of the template as required. Infrastructure Automation automatically prefixes *Copy of* to the old template **Name**.

   Note:

   To create a copy of an orchestration template into a new template, the old and new template content must differ.
4. Click **Add**.

Deleting Orchestration Templates

Complete the following procedure to delete orchestration templates.

1. Navigate to **Services > Catalogs** and select **Orchestration Templates** in the accordion menu.
2. Select the orchestration template you want to delete from the **All Orchestration Templates** list.
3. Click **Configuration**, then click **Remove this Orchestration Template from Inventory**.
4. Click **OK**.

Note:

Read-only templates cannot be deleted.

Retirement

Retiring Virtual Machines

Retiring Virtual Machines and Instances

When a virtual machine or instance is no longer required, it can be retired. Once a virtual machine or instance reaches its retirement date, it is immediately shut down and not allowed to restart. If an attempt to restart is made, Infrastructure Automation will shut down the virtual machine or instance.

There are three built-in policies involved with retirement:

- If the virtual machine or instance reaches its retirement date, it will be stopped even if it is running.
- If a retired virtual machine or instance is requested to start through Infrastructure Automation, the virtual machine or instance will not be allowed to start.
- If a provider starts a retired virtual machine or instance outside of Infrastructure Automation, the virtual machine or instance will be stopped.

Infrastructure Automation provides a number of ways to retire a virtual machine or instance:

- By using the allocated buttons in the Infrastructure Automation console.
- When creating a provision request, a retirement date can be set up.

Using the Console to Retire a Virtual Machine

Through the Infrastructure Automation console, you can retire a virtual machine on a specific date or immediately.

Retiring a Virtual Machine Immediately

1. Navigate to **Compute > Infrastructure > Virtual Machines**.
2. Select the virtual machine or instance that you want to retire.
3. Click (**Lifecycle**), then (**Retire this VM/Instance**).

The virtual machine or instance is immediately stopped, and will be shut down if an attempt is made to restart it.

Setting a Retirement Date and Time for a Virtual Machine or Instance

You can schedule virtual machine retirement by specifying a date and time, or by selecting a relative time a number of months, weeks, days or hours ahead of the present time.

1. Navigate to **Compute > Infrastructure > Virtual Machines**.
2. Select the virtual machine or instance that you want to set a retirement date for.
3. Click (**Lifecycle**), then (**Set Retirement Dates**).
4. From **Enter Retirement Date as**, select **Specific Date and Time** or **Time Delay from Now** to schedule retirement.
    1. To choose a **Specific Date and Time**, click the **Retirement Date and Time** field to open the calendar.
        1. Select a retirement date using the calendar control.
        2. Click then select a retirement time (in UTC) using the arrows.
    2. To retire the virtual machine using a relative time, select **Time Delay from Now**.
        1. From **Time Delay**, specify a retirement time any number of months, weeks, days, or hours in the future using the arrows.
5. Select a **Retirement Warning** if desired.
6. Click **Save**.

The scheduled retirement date and time display in the virtual machine summary screen.

Removing a Retirement Date for a Virtual Machine or Instance

1. Navigate to **Compute > Infrastructure > Virtual Machines**.
2. Select the virtual machine or instance that you want to remove the retirement date from.
3. Click (**Lifecycle**), then click (**Set Retirement Date**).
4. Save a blank date to remove the retirement dates.

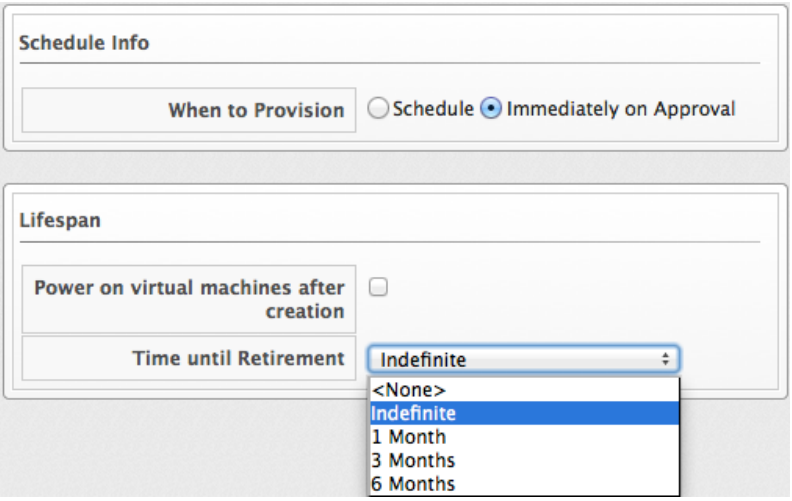
Setting Retirement in a Provision Request

If you are using Infrastructure Automation to provision, you can set when you want retirement in the provision request. To see how to create a request, see Provisioning Requests. A warning email will be sent to the owner before the retirement.

Scheduling Retirement in a Provision Request

When provisioning a cloud instance or virtual machine, a multi-tabbed screen appears where you can set up your provision requests.

1. Click the **Schedule** tab to set when to provision your request and the lifespan of the virtual machine or instance.
2. In **Lifespan**, you can choose to power on the virtual machines or instances after creation and set the **Time until Retirement**. If you select the time until retirement, you will select **Retirement Warning** accordingly.
3. Click **Submit**.



Extending Retirement Dates

Infrastructure Automation **Automate** includes a method to extend the retirement of a virtual machine or instance by 14 days. This section describes how to create a button that invokes this method and how to edit the method to change the number of days.

Creating a Custom Button to Extend Retirement

1. Navigate to **Automation > Embedded Automate > Customization**.
2. Click the **Buttons** accordion.
3. From the **Object Types** tree, select **VM and Instance**.
4. Navigate to the button group to which you want to add this button. (If you do not have a button group, add one and then create the button.)
5. Click (**Configuration**), then (**Add a new Button**).
6. Type in a button text and button hover text, and select the image you want to use.
7. In **Object Details**, select **Request** from the **/System/Process/** dropdown. By default, the message is `create`. Do not change it.
8. In **Request**, type `vm_retire_extend`.
9. Click **Add**.

Changing the Number of Days to Extend Retirement

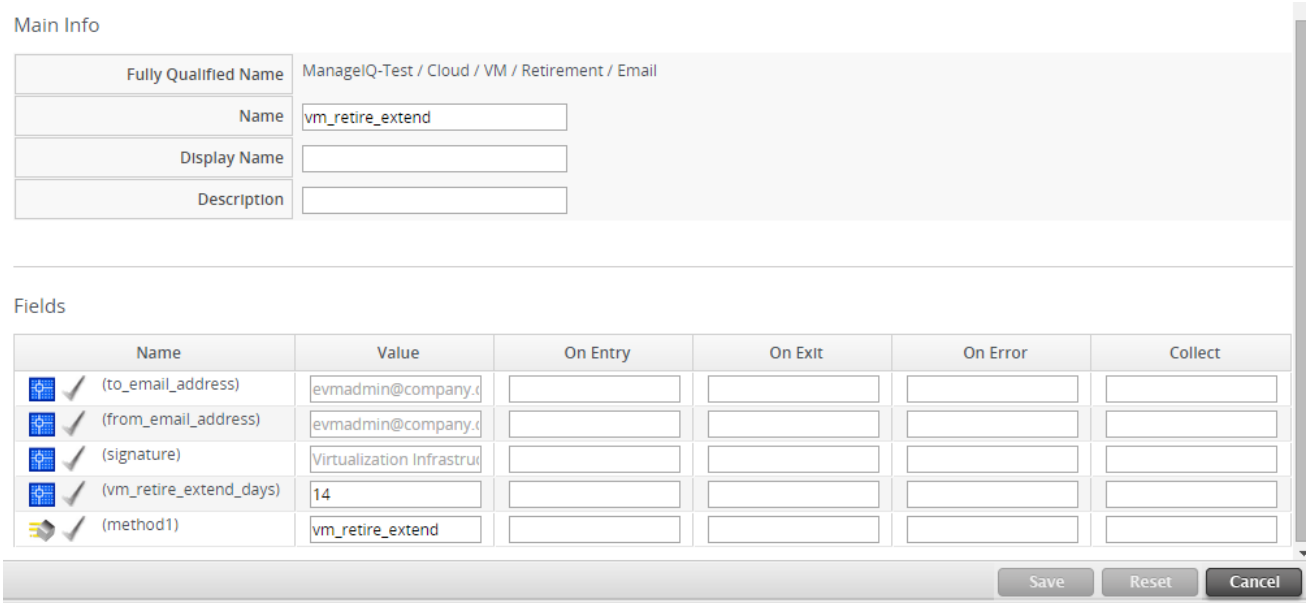
1. Navigate to **Automation > Embedded Automate > Explorer**.
2. Click **DOMAIN > Cloud > VM > Retirement > Email > System > Request**.

   Note:

   DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.

   This example uses the **Cloud** Namespace, but you can also use the **Infrastructure** namespace.
3. Click (**Configuration**), then (**Edit this Instance**).
4. In the Value field, change the **vm_retire_extend_days** attribute to the new value.

5. Click **Save**.



- Creating a Service in for Virtual Machine Provisioning
  - Overview
  - Examining the Provisioning Dialog
  - Creating a Service Dialog
  - Creating a Catalog Item
  - Provisioning a Virtual Machine Using the Self Service User Interface

Creating a Service in for Virtual Machine Provisioning

Overview

Infrastructure Automation provides the ability to define services, which can include one or more virtual machines or instances, and can be deployed across hybrid environments. Services can be made available to users through the Self Service user interface, allowing users to order predefined IT services without IT operations getting involved.

The following three items are required to make a service available to users from the Infrastructure Automation Self Service user interface:

1. **Service Dialog**: A service dialog is where you allow users to configure virtual machine or instance options. A service dialog determines which options users get to change. You can choose the options presented to the user, from setting the service name to changing all provisioning dialog options.
2. **Service Catalog**: A service catalog is used to group service dialogs together.
3. **Service Catalog Item**: A service catalog item is the actual service which combines a service dialog with a provisioning dialog.

Examining the Provisioning Dialog

Understanding provisioning dialogs is important when working with services in Infrastructure Automation. Every provider in Infrastructure Automation comes with a sample provisioning dialog covering the options specific to that provider. The process begins with the provisioning dialog, which presents the basic configuration options for a virtual machine or instance. The options presented are dependent on the provider you are using. For instance, a cloud provider might have flavors of instances, whereas an infrastructure provider might allow you to set the memory size or number of CPUs on a virtual machine.

To see the sample provisioning dialog for Red Hat Virtualization:

1. Browse to menu: **Automation > Automate > Customization**.
2. Expand the **Provisioning Dialogs** accordion and select **Sample RedHat VM Provisioning Dialog** under **VM Provision**.

This is a textual representation of the dialog you will get when you provision a virtual machine.

Creating a Service Dialog

Using a service dialog you can determine which options users get to change. When users access a service, the majority of options available to them are preset and cannot be altered. A service dialog exposes some of those options to the user so that even if they are ordering a basic RHEL 7 machine, for example, they can at least choose the amount of memory, virtual CPUs, or other options available to the instance they order. In cases where certain fields must be unique, such as the name of virtual machines in Red Hat Virtualization, users must enter their own unique name for the virtual machine they choose or the operation will fail, so this field must be exposed.

Note:

A service requires an assigned service dialog, regardless of exposing details to user. A simple service dialog with a **Submit** and **Cancel** button is required at minimum.

A service dialog contains three components:

- One or more **Tabs**.
- Inside the tabs, one or more **Sections**. Note that in the previous method of creating a service dialog using the Infrastructure Automation user interface, **Sections** were referred to as **Boxes**.
- Inside the sections, one or more **Elements**. Elements are controls that accept input. Elements contain methods, like check boxes, drop-down lists or text fields, to fill in the options on the provisioning dialog.

The names of the elements must correspond to the options used in the provisioning dialog.
The following procedure outlines how to create a service dialog for your sample service:

1. Browse to menu: **Automation > Automate > Customization**.
2. Click the **Service Dialogs** accordion.
3. Click **Configuration**, and then (**Add a new Dialog**).
4. Enter basic information for your dialog under **General**:
    1. Enter `Provision RHV VM` in **Dialog's name**, and enter a description for the dialog in **Dialog's description**.
5. Add a new tab to the dialog:
    1. Click **Create Tab**. Then, click the icon on the new tab to edit tab information.
    2. Enter `General` in **Label**.
    3. Optional: Enter a description for the tab in **Description**.
    4. Click **Save**.
6. Add a new section to the tab:
    1. Click **Add Section**. Then, click the icon on the upper-right to edit section details.
    2. Enter `Service and VM Name` in **Label**.
    3. Optional: Enter a description for the section in **Description**.
    4. Click **Save**.
7. Add a **Text Box** element to the section for the service name:
    1. From the list of elements on the left, click the **Text Box** element, then drag-and-drop it inside the section. Then, click the icon next to the element to edit its field details.
    2. Enter `Service Name` in **Label**.
    3. Enter `service_name` in **Name**.
    4. Click **Options**, then select **String** from the **Value Type** list.
    5. Click **Save**.
8. Add a **Text Box** element to the section for the virtual machine name:
    1. From the list of elements on the left, click the **Text Box** element, then drag-and-drop it inside the section. Then, click the icon next to the element to edit its field details.
    2. Enter `VM Name` in **Label**.
    3. Enter `vm_name` in **Name**.

4. Click **Options**, then select **String** from the **Value Type** list.
5. Click **Save**.

9. Add a **Text Box** element to the section for the virtual machine description:
    1. From the list of elements on the left, click the **Text Box** element, then drag-and-drop it inside the section. Then, click the icon next to the element to edit its field details.
    2. Enter `VM Description` in **Label**.
    3. Enter `vm_description` in **Name**.
    4. Click **Options**, then select **String** from the **Value Type** list.
    5. Click **Save**.
10. Add a new section to the tab:
    1. Click **Add Section**. Then, click the icon on the upper-right to edit section details.
    2. Enter `VM Characteristics` in **Label**.
    3. Optional: Enter a description for the section in **Description**.
    4. Click **Save**.
11. Add a **Dropdown** element to the section for the number of CPUs:
    1. From the list of elements on the left, click the **Dropdown** element, then drag-and-drop it inside the section. Then, click the icon next to the element to edit its field details.
    2. Enter `Number of CPUs` in **Label**.
    3. Enter `option_0_cores_per_socket` in **Name**.
    4. Click **Options**. Next to **Entries**, click + to add individual entries for the values `1`, `2`, and `4`, and their description.
    5. Click **Save**.
12. Add a **Dropdown** element to the section for the virtual machine memory:
    1. From the list of elements on the left, click the **Dropdown** element, then drag-and-drop it inside the section.
    2. Click the icon next to the element to edit its field details.
    3. Enter `VM Memory` in **Label**.
    4. Enter `option_0_vm_memory` in **Name**.
    5. Click **Options**. Next to **Entries**, click + to add individual entries for the values `1024`, `2048`, and `4096` and their description.
    6. Click **Save**.
13. Click **Save** to save the dialog.

Your service dialog is now added in the **Service Dialogs** accordion.

Creating a Catalog Item

Create a service catalog item to represent that service in a catalog.

1. Browse to menu: **Services > Catalog**.
2. Click the **Catalog Items** accordion.
3. Click **Configuration**, and then (**Add a new Catalog Item**).
4. Select **RHEV** from the **Catalog Item Type** list.
5. Enter the basic details in the Basic Info tab:
    1. Enter a name for the catalog item in **Name**.

2. Enter a description for the catalog item in **Description**.
3. Select **Display in Catalog**.
4. Select the catalog you created from the **Catalog** list.
5. Select the service dialog you created from the **Dialog** list.

6. Click the **Details** tab:
    1. Enter a description of the catalog item using HTML tags.
7. Click the **Request Info** tab to enter the request details. Enter the same data you did in the traditional provisioning dialog.
    1. In the **Catalog** tab, select the template on which to base the virtual machine from the **Selected VM** area.
    2. Enter `changeme` as the default name of the virtual machine in **VM Name**.
    3. In the Network tab, select **rhevm** from the **vLan** list.
8. Click **Add**.

Note:

In Infrastructure Automation, service catalog items can be more complex than a single virtual machine and are provisioned through a series of automation workflows. The service dialog you created will be used to gather information from the user requesting the virtual machine, hiding all the complexity of the details needed to actually deploy any of the components.

You can now provision a virtual machine through Red Hat Virtualization using the Self Service user interface.

Provisioning a Virtual Machine Using the Self Service User Interface

1. Log in to the Self Service User Interface at https://<hostname/ui/service>.
2. Click the **Service Catalog** tab.
3. Click the service you created.
4. In the **Service and VM Name**:
    1. Enter the service name in **Service Name**.
    2. Enter the virtual machine name in **VM Name**.
    3. Enter the virtual machine description in **VM Description**.
5. In **VM Characteristics**, enter **Number of CPUs** and **VM Memory**.
6. Click **Add to Shopping Cart** and access the shopping cart by clicking its icon on the upper-right corner of the screen.
7. Click **Order**.

You have ordered a new provisioning request. You can follow its progress and see additional details of the order request by selecting it from the **Orders** tab.

Also, see the Self Service User Interface Guide for more information about using the Self Service user interface.

Summary.

As part of the above sample service, you created a provisioning dialog that defines the options that can be set on a virtual machine. Then, you created a service dialog which allows us to expose certain options to be set by the user. For our example, the service name, virtual machine name, virtual machine description, and virtual machine characteristics are configurable. Additionally, you created a service catalog and finally a catalog item. The catalog item combines the service dialog with all of the options in the provisioning dialog. Lastly, you provisioned a Red Hat Virtualization virtual machine using the Self Service user interface.

- Service Provisioning Using an Orchestration Template
    - Overview
    - Creating an Orchestration Template
    - Creating a Service Dialog from an Orchestration Template
    - Creating an Orchestration Catalog Item
    - Creating a Catalog
    - Ordering a Service
    - Orchestration Stacks

Service Provisioning Using an Orchestration Template

Overview

Cloud orchestration is a service that allows you to create, update, and manage cloud resources and their software components as a single unit and then deploy them in an automated, repeatable way through a template. The deployed virtual machine instances and associated collection of resources are referred to as stack. Infrastructure Automation supports a number of orchestration template types, including Amazon CloudFormation and OpenStack Heat, that make deploying complex services easier in the cloud.

In the following example workflow, you will create an orchestration template, and add it as a catalog item to a service catalog. You can then create an orchestration stack from the template, and launch the stack from the catalog using a service dialog.

Creating an Orchestration Template

Complete the following procedure to create a new orchestration template.

1. Navigate to **Services > Catalogs** and select **Orchestration Templates** in the accordion menu.
2. Click **Configuration**, then click **Create a new Orchestration Template**.
3. Enter a **Name** and **Description** for your template.
4. Select **Amazon CloudFormation** from the **Template Type** list.
5. Select **Draft** to create a draft template.
6. Add your template in the area below for the selected **Template Type**. You can author your own stack template, or you can copy and paste from an existing text file.
7. Click **Add**.

Next, create a service dialog from the orchestration template.

Creating a Service Dialog from an Orchestration Template

Complete the following procedure to create a service dialog based on the input parameters defined in the orchestration template.

1. Navigate to **Services > Catalogs** and click **Orchestration Templates** in the accordion menu.
2. Expand **All Orchestration Templates**, then click the orchestration template you created in Creating an Orchestration Template to create a service dialog from it.
3. Click **Configuration**, then click **Create Service Dialog from Orchestration Template**.
4. Enter a name for the service dialog in **Service Dialog Name**.
5. Click **Save**.

Creating an Orchestration Catalog Item

Complete the following procedure to create a new catalog item from the orchestration template you created in Creating an Orchestration Template.

1. Navigate to **Services > Catalogs** and select **Catalog Items** in the accordion menu.
2. Click **Configuration**, then click **Add a New Catalog Item**.
3. Select **Orchestration** from the **Catalog Item Type** list.
4. Enter the basic details in the **Basic Info**:
    1. Enter a **Name** and **Description** for the new service catalog item.
    2. Select **Display in Catalog** box.
    3. Select the appropriate catalog from the **Catalog** list.
    4. Select the appropriate dialog from the **Dialog** list.

5. Select the orchestration template you created in Creating an Orchestration Template from the **Orchestration Template** list.
6. Select **Amazon** from the **Provider** list.

5. Click **Add**.

Next, assign the catalog item to a new service catalog.

Creating a Catalog

Complete the following procedure to create a new catalog, and assign the catalog item you created in Creating an Orchestration Template.

1. Navigate to **Services > Catalogs** and select **Catalogs** in the accordion menu.
2. Click **Configuration**, then click **Add a New Catalog**.
3. In **Basic Info**, add **Name** and **Description** for the new catalog.
4. Under **Assign Catalog Items**, select the catalog item you created in Creating an Orchestration Template from the **Unassigned** box, then move it to the **Selected** box.
5. Click **Add**.

Ordering a Service

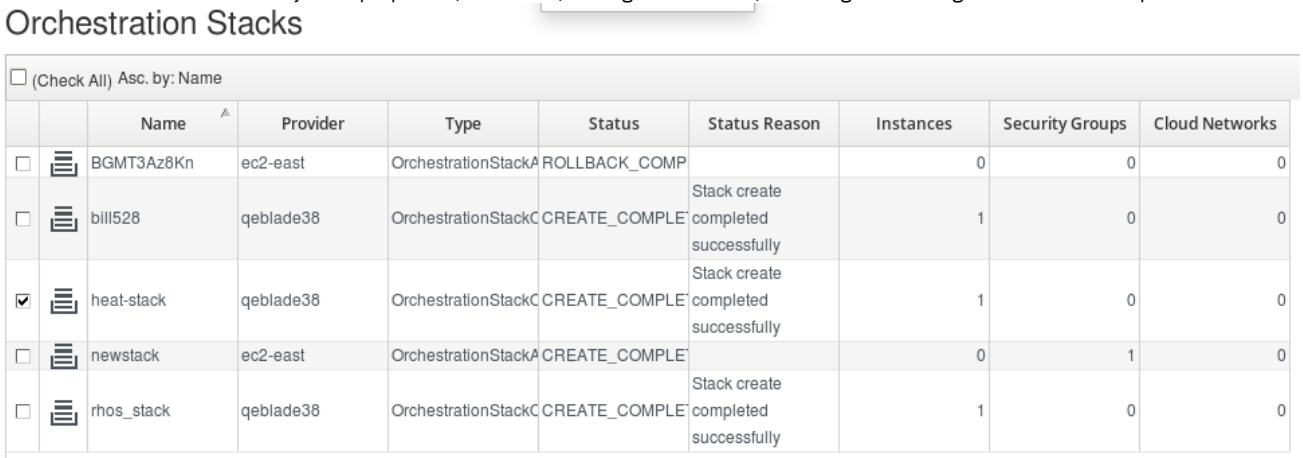
Complete the following procedure to order catalog items from the service catalog.

1. Navigate to **Services > Catalogs** and select **Service Catalogs** in the accordion menu.
2. From **All Services**, select the catalog and the services (catalog items) that you want to order from your catalog, then click **Order**. You will see the **Order Service** window with **Options** and **Parameters**.
3. Enter a name for the stack in **Stack Name**.
4. From the **On Failure** list, select what to do if stack creation fails. The default is **Rollback**.
5. Optional: Enter **Timeout** in minutes.
6. Set the remaining options as needed that will vary depending on the dialog.
7. Click **Submit**.

You have now submitted your service provisioning request. Once a request has been approved, the various stages of fulfillment will be processed. You can monitor the request status and other details in **Services > Requests**.

Orchestration Stacks

1. Once the status of the provisioning request in **Services > Requests** is **Finished**, click **Compute > Clouds > Stacks** to see the deployed stack.
2. Click the stack to see a summary of its properties, resources, among other details, including the running instances that are part of the stack.



You have now deployed instances and associated collection of resources (referred to as a stack) using an orchestration template.

- Assigning a Custom Analysis Profile to a Virtual Machine
  - Creating a Virtual Machine Analysis Profile
  - Creating an Action to Assign the Virtual Machine Analysis Profile to the Analysis Task
  - Creating a Virtual Machine Control Policy
  - Creating a Policy Profile and Assigning the Virtual Machine Control Policy
  - Assigning the Policy Profile to a Virtual Machine

Assigning a Custom Analysis Profile to a Virtual Machine

You can create and assign different analysis profiles for specific virtual machine configurations using the Infrastructure Automation user interface. This document describes the steps required to create a custom virtual machine analysis profile, and assigning it to a virtual machine for use with SmartState analysis, via a control policy.

1. Creating a Virtual Machine Analysis Profile.
2. Creating an Action to Assign the Virtual Machine Analysis Profile to Analysis Task
3. Creating a Virtual Machine Control Policy.
4. Creating a Policy Profile and Assigning the Virtual Machine Control Policy.
5. Assigning the Policy Profile to a Virtual Machine.

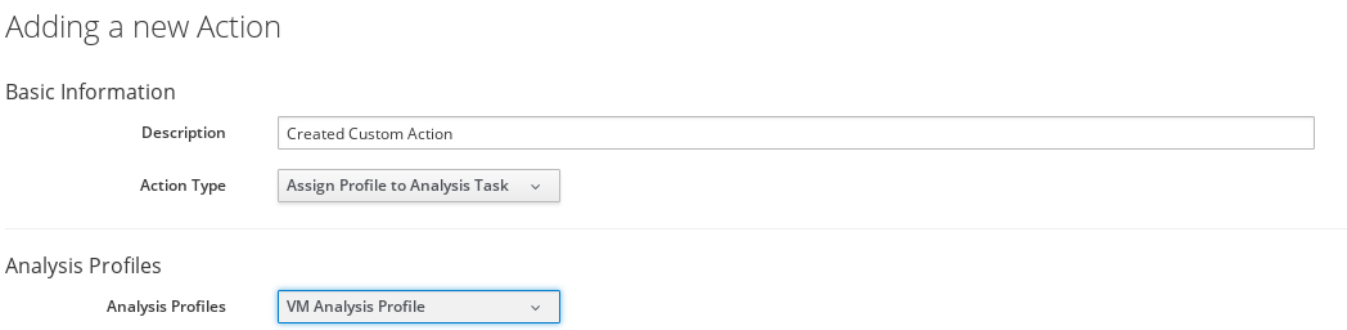
Creating a Virtual Machine Analysis Profile

1. Log into the appliance as the **admin** user.
2. Click **Configuration**.
3. Expand the **Settings** accordion, then click **Analysis Profiles**.
4. Click (**Configuration**), then (**Add VM Analysis Profile**).
5. In **Basic Information**, enter **Name** and **Description** for the analysis profile.
6. In the **Category** tab, select the categories you want to collect information for under **Category Selection**. The **Category** tab is available for virtual machine profiles only.
7. In the **File** tab, click **<New Entry>** and specify the file to be scanned and if you want to collect contents for, then click **Save**. Repeat the step for adding multiple file name entries.
8. In the **Registry** tab, enter the **Registry Key** and **Registry Value**, then click **Save**. To determine whether a registry key exists without providing a value, enter ` ` in *Registry Value</emphasis>. The **Registry** tab is available for virtual machine profiles only.
9. In the **Event Log** tab, specify the event log entries to collect. Enter a **Filter Message** to look for specific text in a message. Enter a **Level** (*info*, *warn* or *error*, for example) to specify the event level. Enter the **Source** for the event log entry and **# (Number) of Days** to specify how far back to check.
10. Click **Add**.

Creating an Action to Assign the Virtual Machine Analysis Profile to the Analysis Task

Actions are performed after the condition is evaluated. You can associate actions with specific events when you create a policy. Infrastructure Automation provides a set of default actions, but you can also create custom actions using the Infrastructure Automation user interface.

Use this procedure to create a custom action by adding the **Assign Profile to Analysis Task** action type to the virtual machine analysis profile (created in Creating a Virtual Machine Analysis Profile).



1. Browse to menu: **Control > Explorer**.
2. Expand the **Actions** accordion and click **Configuration**, then (**Add a new Action**).
3. Enter a **Description** for the new action. This will be the name given to your new action.
4. Select **Assign Profile to Analysis Task** from the **Action Type** list.
5. Select the newly-created virtual machine analysis profile from the **Analysis Profiles** list.
6. Click **Add**.

Note:

You can only associate this action with an analysis start event.

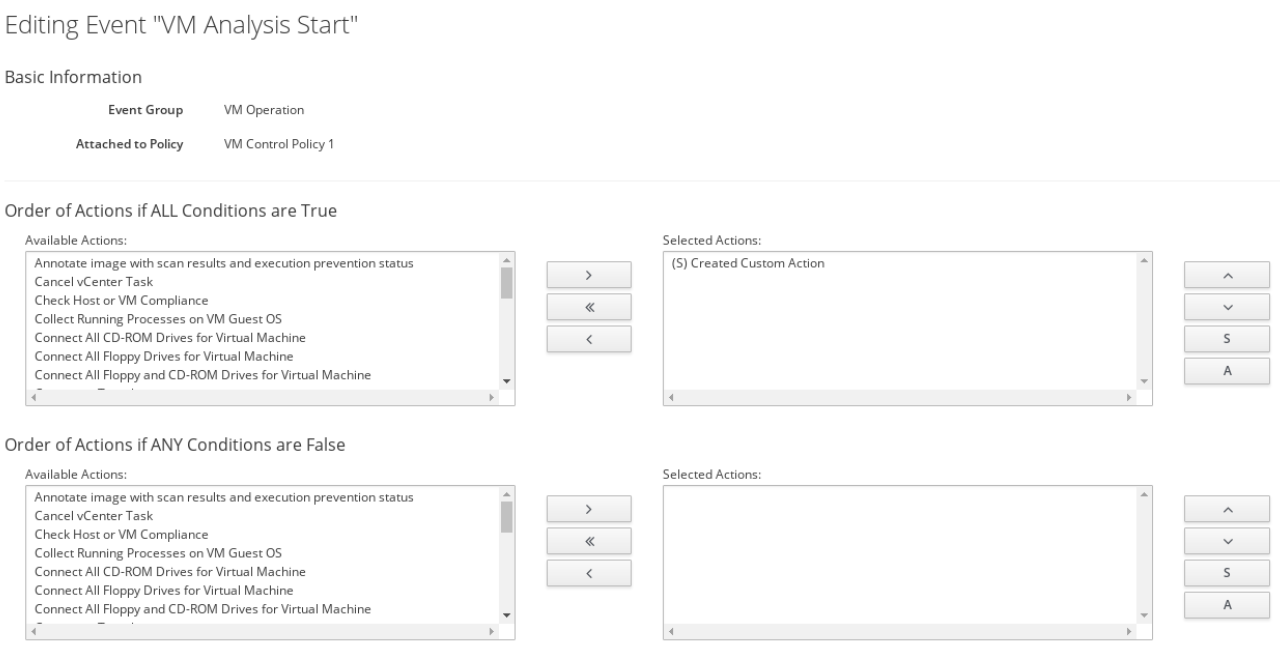
The action is created and added to the **Available Actions** list. Associate this action with the **VM Analysis Start** event when you create a virtual machine control policy in the next procedure.

Creating a Virtual Machine Control Policy

You can create a control policy by combining an event, a condition, and an action. The procedure below describes how to create a virtual machine control policy to assign the newly-created action to the **VM Analysis Start** event. Optionally, you can use a scope expression that is tested immediately when the policy is triggered by an event. If the item is out of scope, then the policy will not continue on to the conditions, and the assigned action will not run.

1. Browse to menu: **Control > Explorer**.
2. Expand the **Policies** accordion, and click **Control Policies**.
3. Select **Vm Control Policies**.
4. Click **Configuration**, then **Add a New VM and Instance Control Policy**.
5. Enter a **Description**. This will be the name given to your VM control policy.
6. Clear the **Active** box if you do not want this policy processed even when assigned to a resource.
7. Optional: Enter a **Scope** (you can also create a scope as part of a condition, or not use one at all). If the virtual machine is not included in the scope, the assigned action will not run.
   - You can use the drop-down list to create an expression for the **Scope**. Based on what you choose, different options appear. Click **Commit expression element changes** to add the scope.
8. Enter **Notes** if required.
9. Click **Add**. The policy is added and listed under **Vm Control Policies** in the **Policies** accordion.
10. Select the newly-added VM control policy. You can now associate events, conditions, and actions with the policy.
11. Click **Configuration**, then **Edit this Policy's Event assignments**.
12. Under **VM Operation**, set **VM Analysis Start** to **Yes**.
13. Click **Save**.
14. Click the **VM Analysis Start** event to configure actions.

15. Click **Configuration**, then **Edit Actions for this Policy Event**.
16. In **Order of Actions if ALL Conditions are True**, select the action created in Creating an Action to Assign the Virtual Machine Analysis Profile to the Analysis Task from the **Available Actions** list. This action will take place if the resources meet the conditions of the policy.



Note:

Each selected action can be executed synchronously or asynchronously; a synchronous action will not start until the previous synchronous action is completed, while an asynchronous action allows the next action to start whether or not the first action has completed. Also, at least one Infrastructure Automation server in the Infrastructure Automation zone must have the notifier server role enabled for the trap to be sent.

17. Click ( ) which will move the action to **Selected Actions**. The selected action is set to (S) Synchronous by default. From **Selected Actions**, select the action, then:
    - Click **A** (Set selected Actions to Asynchronous) to make it asynchronous.
    - Click **S** (Set selected Actions to Synchronous) to make it synchronous. If creating a synchronous action, use the up and down arrows to identify in what order you want the actions to run.
18. Click **Save**.

Creating a Policy Profile and Assigning the Virtual Machine Control Policy

Add a new policy profile and assign the virtual machine control policy (created in Creating a Virtual Machine Control Policy) to it.

1. Browse to menu: **Control > Explorer**.
2. Expand the **Policy Profiles** accordion, click **Configuration**, then **Add a New Policy Profile**.
3. In the **Basic Information** area, enter a **Description** for the policy profile.
4. Under **Policy Selection**, select the virtual machine control policy created in Creating a Virtual Machine Control Policy from the **Available Policies** list.
5. Click ( ) to move the selected virtual machine control policy into this profile.
6. Click **Add**.

Assigning the Policy Profile to a Virtual Machine

Assign the policy profile you created in Creating a Policy Profile and Assigning the Virtual Machine Control Policy to a virtual machine, and initiate SmartState analysis.

Note:

Policy profiles can be specified at multiple levels. If you assign a policy profile to a provider (Amazon EC2 or OpenStack for example), the profile will apply to all hosts or virtual machines for that provider.

1. Browse to menu: **Compute > Clouds > Instances** or menu: **Compute > Infrastructure > Virtual Machines**. Select the virtual machine or instance.
2. Click **Policy**, then **Manage Policies**.
3. Under **Select Policy Profiles**, select the policy profile created in Creating a Policy Profile and Assigning the Virtual Machine Control Policy. It will turn blue to show the selection. Click the triangle next to the policy profile to see its member policies.
4. Click **Save**.
5. Click **Configuration**, then **Perform SmartState Analysis**. A pop-up window appears to confirm the action.
6. Click **OK**. SmartState analysis is initiated for the selected virtual machine or instance from the Infrastructure Automation database.

SmartState analysis will now report back findings specified by the custom virtual machine analysis profile.

Creating a tag mapping for Infrastructure management integration

It is possible to use the virtual machines created by the Service library in the Infrastructure management. This is required for any additional tags defined by the user and used for pricing and policy integration. For example, if policy or pricing requires any additional tags, they need to be created in the Managed services in service definition and also mapped in Infrastructure management.

The tag mapping requires the following steps on the Infrastructure management side:

1. Create a tag mapping of the `environment` tag category. See Tag mapping.
2. Refresh the configuration. See Refreshing Cloud Providers.
3. Verify whether the virtual machine is tagged correctly by using the following steps:
    1. On the Infrastructure management console, click **Compute**>**All VMs**.
    2. Select the virtual machine.
    3. Verify the tags in the **Smart Management** section.
    4. The virtual machines provisioned by the Managed services should have a list of all mapped labels.

Tag mapping is supported for the following cloud providers:

- Amazon
- Microsoft Azure
- Kubernetes
- OpenShift
- OpenStack

Provisioning templates

Learn more about provisioning templates for Infrastructure Automation.

- Managing shared parameters
- Creating and managing templates
- Creating and managing template instances

Managing shared parameters

Shared parameters define shared configuration to store data and are used by templates.

The Data Types and Data Objects are constructs that are used to capture these shared configurations. These are essentially pointers to data structures that facilitate the dynamic binding at run time. You can create and manage Data Types and Data Objects in Managed services console. In addition, whenever you import a template from Git repository, the Data Types of the template are also imported.

Data Types

The Data Types defines the data. Data Types contains the metadata of the attributes like name, attribute type and default values. For example, you can have a data type **IBM_Tivoli_Directory_Structure** to encapsulate the attributes of your LDAP server and then you can create a data object, an instance of this data type, that contains the values for these attributes which can be referenced in the template during deployment.

There are two types of Data Types:

- Explicit Data Types

  An administrator creates an explicit datatype either through a REST call or from the Admin Console UI. To create Data Type from the user interface, see Managing Datatypes from the user interface. To create Data Type by using REST APIs, see Shared parameters APIs. The name of the Data Types must be unique and for explicit data types the name must contain only alphanumeric and "_-" characters, and must start with an alphanumeric character.

- Implicit Data Types

  As Data Types and Data Objects are pointers to data structures, most of the existing constructs in Managed services can be referenced through an implicit Data Types. A use case for implicit data type is creation and use of a new cloud connection on the run without having to pre-define the cloud connection. This is very powerful when you have a service composed of multiple templates, and one of the templates depends on the cloud connection created by the template run prior to it. A cloud connection type can be referenced by **com.ibm.cloud.cloudconnections** followed by the provider name as follows **com.ibm.cloud.cloudconnections.mycloud**. Any data object instance of this datatype, for example, **com.ibm.cam.provider.mycloud.dev_acccount** points to this cloud connection.

  You can define these implicit data types and objects in the **output_datatypes** section of camvariables.json file as follows in your template. On deployment of the template a new cloud connection **com.ibm.cloud.cloudconnections.mycloud.dev_acccount** with atttributes **username**, **password** and **server** with their correspoding values will be created and can be used by other templates.

```
"output_datatypes": [
  {
    "name": "com.ibm.cloud.cloudconnections.mycloud",
    "dataobject": {
      "name": "dev_acccount",
      "attributes": [
        {
          "name": "username",
          "value": "${template_input_params.admin_user}"
        },
        {
          "name": "password",
          "value": "${template_input_params.admin_password}"
        },
        {
          "name": "server",
          "value": "${template_output_params.private_master_ip}"
        }
      ]
    }
  }
],
```

Data Types lookup

Data Types lookup is the mechanism to reference a Data Type to retrieve the dataobjects associated with it. The API representation is **POST datatypes/lookup** wherein the Data Type is passed as a parameter in the body. For more information, see Shared parameters APIs.

Data Objects

The Data Objects represent an instantiation of a Data Type. For example, if you have a data type **IBM_Tivoli_Directory_Structure** that encapsulates your LDAP server, then you can create the following instances **IBM_Tivoli_Directory_Structure** of Data Type to represent the LDAP server for different environments.

- **Development_IBM_Tivoli_Directory_Structure**
- **Test_IBM_Tivoli_Directory_Structure**
- **Production_IBM_Tivoli_Directory_Structure**

The Data Objects inherit the attributes of the parent Data Type. If no value is provided for an attribute, the default value from the Data Type definition is used.

To create Data Object from the user interface, see Managing Dataobjects from the user interface. For more information about the REST APIs, see Shared parameter APIs.

Create Data Types and Data Objects from user interface

The following links provide the steps to create Data Types and Data Objects from the **Manage** > **Shared parameters** tab of the user interface:

- Data Types
- Data Objects
- Prebuilt Data Type list

Import Data Types from Git repository

To import Data Types, the template must have a datatypes folder with JSON and readme files. If the JSON is not properly defined, then the Data Types are not imported. To resolve this issue, fix the template and reimport the template. If the Data Types are already defined, then they are not overwritten during the template import.

Managing Data Types from the user interface

You can create, edit, and delete Data Types from the **Shared parameters** tab of the user interface.

- Create Data Type
- Upload Attributes
- Add Attributes
- Edit Attributes
- Delete Attributes
- Edit Data Type
- Delete Data Type

Create Data Type

1. Go to **Manage** > **Shared parameters** tab.
2. In the **Data types** section, click **Create data type**.
3. In the **Create data type**, enter the **Name**, **Data type**, and **Description**.

   Notes:

   - The Data Type name must begin with an alphabet and must have a combination of only these characters: [a-zA-Z][a-zA-Z0-9_-]
   - Do not use the following reserved names while naming a Data Type:
     - Anything that starts with `com.ibm.cloud.cloudconnections`
4. Click **Create**. The **Edit data type** page is displayed wherein you can edit name, Data type, and description or work on the attributes.

Upload Attributes

If the Data type has associated Data objects, then only **View attributes** option is available.

1. In the **Data types** section, select **Edit** in the **Actions** overflow menu.
2. Click **Upload attributes**.
3. Click **Import from file** to import from a JSON file or copy and paste in the editor.
4. Click **Upload**.

Add Attribute

If the Data Type has associated Data Objects, then only **View attributes** option is available.

1. In the **Data type** section, select **Edit** in the **Actions** overflow menu.
2. In the **Edit data type** page, click **Upload attributes**.
3. Click **Add attribute**.
4. Enter the **Parameter name**.
5. Select **End-user permission**. The available permissions are **Read-Only** and **Read & Write**.

6. Select a value from the **Parameter type** drop-down list. The available parameter types are string, password, list, boolean, counter, and map.
7. Enter the parameter values and click **Add**:
    - Enter the following common fields for all **Parameter type**:
        - **Display name** - Enter the label of the field.
        - **Description** - Enter a description of the field.
        - **Required** - Select if the field is mandatory. It is available only for **Read & Write** permission.
    - Enter the following values for string **Parameter type**:
        - **Specify options** - The **Specify options** is available for a **Read & Write** permission of type string. If you select **Specify options**, then **Default label** and **Value** pairs are displayed. To add more pairs of **Default label** and **Value**, click **Add**. Click the cross icon of a label to delete it.
        - **Validation rule** - Enter the **Validation rule**.
        - **Default string value** - Enter the default value of the string parameter. It is mandatory for **Read-Only** permission.
        - **Secured** - Select if the field is secured.
    - Enter the following values for map **Parameter type**:
        - **Key** and **Value** pair - Enter the key value pair. To add more key value pairs, click **Add**. Click the cross icon of a key to delete it.
    - Enter the following values for counter:
        - **Default counter value** - Enter the default value for the counter parameter. It is mandatory for **Read-only** permission.
        - **Maximum value** - Enter the maximum value of the counter parameter. It is available only for **Read & Write** permission.
        - **Minimum value** - Enter the minimum value of the counter parameter. It is available only for **Read & Write** permission.
    - Enter the following value for Boolean:
        - **Default boolean value** - Select **True** or **False** for the default Boolean parameter value.
    - Enter the following values for list:
        - **Default list value** - Click **Add** to add default value to the list. Click the cross icon for a default value to delete it.
    - Enter the following values for password:

      Note: Password type is applicable only for **Read & Write** permission.

Edit Attributes

You cannot edit the attributes if the data type is associated with data objects.

1. In the **Edit data type** page > **Attribute** section, select **Edit** in the **Actions** overflow menu for the attribute you want to modify.
2. Update the values and click **Edit** to save the changes.

Delete Attributes

You cannot delete the attributes if the data type is associated with data objects.

1. In the **Edit data type** page > **Attribute** section, select **Delete** in the **Actions** overflow menu for the attribute you want to delete.
2. Click **Delete** in the confirmation message box to delete.

Edit Data Type

You can edit description and upload or add attributes, but cannot edit the **Name** or **Type**. You cannot edit any of the Data Type values if it is associated with Data Objects.

1. In the **Data types** section, search for your Data Type.

   Note: To search for a Data Type, enter the keywords or name of the Data Type.
2. In the **Data type** section, select **Edit** in the **Actions** overflow menu of the Data Type you want to edit.
3. In the **Edit data type** page, click the edit icon to modify the **Description**.
4. In the **Attributes** section, you can add new attributes, edit, or delete an existing attribute.
5. Click **Save**.

Delete Data Type

You cannot delete a Data Type if it has associated data objects.

1. In the **Data types** section, search for your Data Type.

   Note: To search for a Data Type, enter the keywords or name of the Data Type.

2. In the **Data type** section, select **Delete** in the **Actions** overflow menu of the Data Type you want to delete.
3. If there are any associated data objects, a message box lists them. Delete those data objects first and then delete this Data Type.
4. If there are no associated data objects, click **Delete** in the confirmation message box to delete.

Managing Data Objects from the user interface

Shared parameters allow you to create reusable Data Objects. You can create, edit, and delete Data Objects from the **Shared Parameters** section of the user interface.

- Create Data Objects
- Edit Data Objects
- Delete Data Objects
- Duplicate Data Object

Create Data Objects

1. Go to **Settings** > **Shared parameters**.
2. In the **Data objects** section, click **Create data object**.
3. In the **Create data object** page, select or enter the following values:
   - Select the Data Type from the drop-down list.

     Note: Whenever you use the selected Data Type, ensure that the value of Data Object is not empty.

   - In **Select a namespace** > **Assign access** section, select either of the following values based on your requirement:
     - **Make data object globally accessible (available to all users)** - Data objects created globally are accessible by all users.
     - **Make data object part of a namespace** - Associate the data object with a specific namespace. For this option, you must select a **Namespace** from the drop-down list to which the data object must be assigned. The namespace on the **Shared parameters** page is preselected. You can modify it, if needed.
   - Enter a unique name for the Data Object. It is must be unique within the selected Data Type. If you do not enter an unique name or enter a character outside the permissible range, then an error message appears.

     Note: The Data object name must begin with an alphabet and must have a combination of only these characters: [a-zA-Z][a-zA-Z0-9_-]

   - In the **Attributes** section, enter a value for all the attributes. The attributes depend on the selected datatype.
4. Click **Create**.

Edit Data Objects

You can only edit the description and attribute values.

1. In the **Data objects** section, search for your Data Object.

   Note: You can search for a Data Object by using the following methods:

   - To filter Data Objects based on the namespace, select a namespace from the **All namespaces** drop-down list. The list of Data Objects in the page changes based on the selected namespace. You can also select "All namespaces" to see all Data Objects for which you have access.
   - Enter the keywords or name of the Data Objects in **Search data objects**.
   - You can use the show filter icon next to search text to filter records. The namespace option has **Show globally accessible** option, which refers to those Data Objects that are globally accessible across all namespaces and are not assigned to any specific namespace.
2. Select **Edit** from the **Actions** overflow menu of the Data Object you want to update.
3. Update the details and click **Update**.

   Important: You can change a namespace to globally accessible but cannot assign a different namespace for a Data Object. You can also change a globally accessible namespace to a different namespace but be informed that with this namespace change you may not be able to

perform any operations on existing deployments.

Delete Data Objects

1. In the **Data objects** section, search for your Data Object.

   Note: You can search for a Data Object by using the following methods:

   - To filter Data Objects based on the namespace, select a namespace from the **All namespaces** drop-down list. The list of Data Objects in the page changes based on the selected namespace. You can also select "All namespaces" to see all Data Objects for which you have access.
   - Enter the keywords or name of the Data Objects in **Search data objects**.
   - You can use the show filter icon next to search text to filter records. The namespace option has **Show globally accessible** option, which refers to those Data Objects that are globally accessible across all namespaces and are not assigned to any specific namespace.
2. Select **Delete** from the **Actions** overflow menu of the Data Object you want to delete.
3. Click **Delete** in the confirmation message box to proceed with deletion.

Duplicate Data Object

1. In the **Data objects** section, search for your Data Object.
2. Select **Duplicate** from the **Actions** overflow menu of the Data Object.
3. Select either of the following options in **Assign access** section:

   - **Make data object globally accessible (available to all users)** - Data Objects created globally are accessible to all users.
   - **Make data object part of a namespace** - Associate the Data Object with a specific namespace. For this option, you must select a Namespace from the drop-down list to which the template must be assigned.
4. Change the name of the **Data Object Name*. The default name is `Copy_of_<name_of_original_Data_Object`.
5. Click **Duplicate**.

Prebuilt Data Type list

Managed services provides a list of preloaded data types that capture all attributes, for example, resource pool, datacenter, template name, virtual machine domain name of a vSphere cloud. Managed services Templates can make use of these data types and hardcode their values rather than redefining them everytime. This feature simplifies your data entry experience which is otherwise voluminous and repetitive in nature.

To know more about the preloaded data types, see Git repository.

To view the list of data types in Managed services user interface, go to **Manage** > **Shared parameters** tab > **Data types** section. For more details about working with Data Types and Data Object from Managed services user interface, see Managing shared parameters.

Creating and managing templates

Create your templates and deploy the template instances to your environment. Manage helm charts from IBM Cloud Pak for AIOps release management page.

- Defining the structure of Managed services template
- Tags module
- Importing a template
- Editing a template
- Duplicating a template
- Deleting a template
- Viewing a template
- Changing preloaded content tokens

Defining the structure of Managed services template

The Managed services supported Terraform template is made up of datatypes, modules, metadata files and `.tf` files. This section provides you with the structure of the template so that you can build a new Terraform template from scratch.

The following block depicts the structure of a template:

```
- License
- README.md
- main.tf
- variables.tf
- output.tf
- camtemplate.json
- camvariables.json
- datatypes/
  - datatype.json
  - ...
- modules
  - moduleA/
    - LICENSE
    - main.tf
    - output.tf
    - scripts
      - script.sh
    - variables.tf
  - moduleB/
    - ...
- examples/
  - exampleA
    - main.tf
  - exampleB
    - main.tf
  - ...
- scripts
  - script.sh
  - ...
```

The following table describes the different components of a template structure:

Component	Description
cam_template_structure	This is the repository name or root folder of the template.
License	The license under which the template is made available.
README.md	This file contains a description of the template, expected usage, and variables required.
main.tf	The main entry point for the template.
variables.tf	Provides information and defaults of variables that are needed to execute the template.
output.tf	Provides information on important variables, which can be queried after deployment through Terraform and Managed services user interface.
camtemplate.json	A metadata file which provides information such as the short/long name, description, version and more for presentation in the Managed services Catalog.
camvariables.json	A metadata file describing how each variable is presented in the Managed services user interface. This creates an improved deployment experience with features such as: optional drop-downs, secure (masking), regex validation and descriptions.
datatypes directory	Directory contains json files to be loaded into Managed services Shared Parameters.
datatypes/datatype.json	Contains information about the Data Types which can be used by templates to facilitate dynamic binding at deployment time.
modules directory	This is optional and only used if nested Modules are included. Nested Modules allow you to break up complexity to provide more flexible configuration options.
modules/module	Directory for a specific nested Module. It must at least include a main.tf, output.tf and variables.tf.
modules/moduleA/main.tf / variables.tf / output.tf	Files are scoped to Module execution only but have the same definition as the preceding definition.
modules/moduleA/scripts directory	Directory contains scripts or files used by the Module.
scripts	Directory contains scripts or files used by the template.
examples	Directory containing an example(s) of additional configurations.

Modules

Modules are Terraform files in a folder. Even the working directory of a Terraform configuration where you run "Terraform apply" is considered as a root Module.

The Terraform Module is achieved by abstracting common configuration as building blocks, which can be reused by you and the rest of the organization. The Terraform configuration is by itself a Module sourcing other common building blocks (Modules) all with the right level of abstraction to enable reuse. The Terraform Modules provide developers with a structure to define reusable building blocks. You can define reusable configuration to meet industry guidelines and/or certify against your organization's unique standards. Large organizations can take advantage of structuring Terraform Modules by specialties like networking, security or by team ownership. When developing Terraform, use the Modules approach.

The following block depicts the structure of a module:

```
- modules
  - moduleA/
    - LICENSE
    - main.tf
    - output.tf
    - scripts
      - script.sh
    - variables.tf
  - moduleB/
    - ...
```

Tags module

For more information about system tags and user tags, see Tags module.

Managed services JSON files

You can create **camtemplate.json** and **camvariables.json** files for your template. It is not mandatory to have them in the template but its presence simplifies the deployment process.

- **camtemplate.json**

  The **camtemplate.json** file contains template metadata. It defines the function and features of the template.

  In the Managed services, the template **Overview** page includes the details defined in this JSON file.

  For example, you can go through this **camtemplate.json** file to understand its structure at - Sample camtemplate.json.

- **camvariables.json**

  The **camvariables.json** file contains metadata that is used in Managed services to augment the support for variables in Terraform. Metadata adds additional parameters for existing variables that are understood and handled by Managed services. Here, define **template_input_params** for all the variables in **main.tf** file.

  When you deploy the template in the Managed services user interface, all these defined variables are displayed. After you enter the values in the user interface, they are passed on to a JSON file which is in turn sent for execution to Terraform. Terraform matches these parameters with the variables in the **.tf** file.

  The structure of **camvariables.json** file is as follows:

  ```
  {
  "terraform_version": "~> 0.11.0",
  "input_datatypes": [{"name": "namespacename", "label": "Formatted Name"},],
  "output_datatypes": [{""},],
  "input_groups":[{},],
  "output_groups":[{},],
  "template_input_params": [{},],
  "template_output_params":[{},]
  }
  ```

  Managed services variables structure to deploy content runtime:

  ```
  {
  "output_datatypes": "content_runtime_chef",
  "template_input_params": [{},],
  "template_output_params":[{},]
  }
  ```

  Managed services variable structure to consume content runtime in a new deployment:

  ```
  {
  "input_namespaces": [{"name": "content_runtime_chef", "label": "Content Runtime: Chef"}],
  "template_input_params": [{},],
  "template_output_params": [{},]
  }
  ```

  The **template_input_params** is defined for every variable and the structure is as follows:

```
"template_input_params": [{
   "name": "The name of the service parameter",
   "label": "The label to be displayed in the user interface for this parameter",
   "description": "A description for this parameter",
   "type": "The type of parameter. For example, "list, string, map",
   "default": "The default value of the parameter",
   "hidden": "Specify if the parameter is not displayed in the user interface. The value must be true or
false",
   "immutable": "Specify if the parameter is read only. The value must be true or false",
   "immutable_after_create": "Specify if the parameter is read only after create (cannot be changed
during an update or plan/apply). The value must be true or false (default)",
   "required": "Specify if the parameter is required. The value must be true or false",
   "secured": "Specify if the parameter value is shown as ********. The value must be true or false",
   "regex": "Define the regular expression - ^[A-Za-z0-9-]{1,52}$. The value you provide must match this
regular expression."
   "group_name": "The group to locate this parameter in when displaying on the UI". For example,
"myGroup",
   "options": [  <--- in place of "default" for a list to select from
   {
     "value": "dc1",
     "label": "My Favorite DC",
     "default": "true"
    },
    {
     "value": "dc2",
     "label": "some other DC"
    }
   ]
      },]
```

The **template_output_params** is defined for every variable and the structure is as follows:

```
"template_output_params":
[{
   "name": "name of the output variable in the template file", output section from Terraform
   "label": "The label to be displayed in the user interface for this parameter",
   "description": "A description for this parameter",
   "type": "The type of parameter. For example: "list, string, map, url",
   "shortname": The abbreviated name for a value (such as the URL shortname like "console link" which
actually links to https://asdf.com/asdf/asdf.html"
   "hidden": "Specify if the parameter is not displayed in the user interface. The value must be true or
false",
   "secured": "Specify if the parameter value is shown as ********. The value must be true or false",
   "return_type": "The type of the parameter that is returned"
   "group_name": "The group to locate this parameter in when displaying on the UI" such as "myGroup",
   },]
```

Datatypes folder

Datatypes are yet another Managed services specific metadata.

Datatypes

Administrator defined connection is required as the templates that are defined in Managed services need access to external systems that are not necessarily backed up by a Managed services instance. The Data Types and Data Objects are two admin constructs that are used to capture these shared configurations.

You must define the datatypes for a template under the datatypes folder or else they get ignored by the import code:

```
{
   "name": "Name of the datatype",
   "label": "Lable of the datatype",
   "description": "Description of the datatype",
   "attributes": [
    {
      "name": "Name of the attribute",
      "type": "type of the attribute",
      "description": "Description of the attribute",
      "label": "Label of the attribute",
      "required": "Is it a mandatory attribute",
      "secured": "Is it a secured attribute",
      "immutable": "Is it immutable",
      "hidden": "Is it hidden",
      "default": "The default value of the attribute",
      "options": Options of the attribute.
    },
    {
      ...
      ...
```

```
    }
  ]
}
```

For a sample datatype, see GitHub repository.

For more information about shared parameters, see Shared parameters.

Terraform files

The template can have **main.tf** and **variables.tf** terraform files.

- **main.tf**

  The main template that gets deployed. It describes all the variables and resource structure that gets deployed on the cloud.

  Example:

```
module "icam_agent_install_linux" {
source = "./modules/icam_agent_install"
ip_address = "${var.ip_address}"
user = "${var.user}"
password = "${var.password}"
private_key = "${var.private_key}"
icam_agent_location = "${var.icam_agent_location}"
icam_agent_location_credentials = "${var.icam_agent_location_credentials}"
icam_agent_source_subdir = "${var.icam_agent_source_subdir}"
icam_agent_installation_dir = "${var.icam_agent_installation_dir}"
icam_agent_name = "${var.icam_agent_name}"
#######
bastion_host        = "${var.bastion_host}"
bastion_user        = "${var.bastion_user}"
bastion_private_key = "${var.bastion_private_key}"
bastion_port        = "${var.bastion_port}"
bastion_password    = "${var.bastion_password}"
#######
}
```

- **variables.tf** - Terraform variables are defined in this file.

  For example:

```
variable "ip_address" { type = "string" description = "IP Address of the host to install the IBM cloud
App Management Agent"}
variable "user" { type = "string" description ="Userid to install the IBM Cloud  App Management Agent"}
variable "password" { type = "string" descritpion = "password of the installation user."}
variable "private_key" { type = "string" description = "Private key of the installation user" }
```

  These variables are referenced in **camvariables.json** file.

Template lifecycle through branches

It is a best practice to use dedicated repository in Git for templates and auxiliary files. Whenever you make a change, create a branch at the repository level and not at the folder level. You can have multiple templates in the same repository and multiple versions of the same template. Whenever changes do not break compatibility with older versions, you can use the same branch and merge the changes. If you have changes that break compatibility, then use a new branch. In Managed services, import the new branch and make it the default. You can run a plan and apply that upgrade instances to the new configuration.

By using the Template Designer, you can manage branches effectively.

Tags module

Tags module is a helper to read the tags and get it in required formats.

System Tags

Managed services provisions these system tags automatically for a template and predefines their values. The tag keys are customizable so as to be compatible with other keys on the provider.

You can decide whether a system tag must be provisioned or not (enabled/disabled). The enable disable provision is available because the providers may or may not support a tag or they may limit the count of tags allowed. These system tags are available at CAM_terraform/cam-

provider-terraform/stacks/stackjobs/Modules/camtags. If you want to include them in your template, set the source value relative to your path as follows:

```
module "camtags" {
  source = "../Modules/camtags"
}
```

To use the tag in a Terraform resource, write your template so you can read, pass, or format them.

For example, AWS needs its tag as a map:

```
output "tagsAsMap" {
   value = "${module.camtags.tagsmap}
}
```

This tags module has various outputs. It reads the tags that Managed services generates and converts it into a map format. For Deployments, set your tags in AWS and point your tags to this output value. Similarly, if you have a IBM Cloud, it takes tag in a list format with key value pairs. It formats the tags in a list and you can pass it to IBM Cloud.

User Tags

You can specify these tags at the time of provisioning. There are many ways a user tag can be designed, associated, and managed with a template.

Note: Consider the provider specific limitations when you name the tag or enter a key for it. For more information about IBM Cloud provider naming conventions, see Working with tags.

Tag Key Name can be set up / established at the following levels:

- Global - If you specify a user tag key here, it shows up whenever you request any template across all users / groups.
- Group - If you specify a user tag key here, it shows up whenever you request a template within a specific group.
- Template - If you specify user tag key here, it shows up whenever you request the specific template.
- Service - A tag key can be setup when you create a service in the Service Composer and it can be specific to service provisioning.

When a user specifies a tag key, they have an option of restricting the value types. The set of value types must be as follows:

- String - Any string possibly with a limit on the length and/or restricting certain characters.
- Integer - Only integers
- Number - Any number including long, decimal, float, and so on.
- Enumerated Value - A specific set of values with no ability to add new ones and the first one is set by default.
- Dynamic Value - Ability to specify a configuration or external API to retrieve the set of values allowed for this tag key dynamically.

For more information about using a template that has tags in a service, see Prerequisites for tags.

Importing a template

You can import a template from your Git repository.

About the task

By default, templates that are imported from Git repositories are not tagged to any namespace and are globally accessible. You can create a clone of the template and then associate a namespace to it.

Procedure

1. In the navigation bar, click **Library** > **Terraform templates**. The template library is displayed.
2. Click **Import template**.
3. For GitHub, GitLab, and Bitbucket Server, select either of the following options in **Assign access*** section:
   - **Make template globally accessible (available to all users)** - Templates created globally are accessible by all users.
   - **Make template part of a namespace** - Associate the template with a specific namespace. For this option, you must select a **Namespace** from the drop-down list to which the template must be assigned.

   Note:
   - If you create a template in a specific namespace, then only deployments to that namespace can use the template.
   - Based on the value selected in the namespace drop-down list of the **Library** > **Terraform templates** main page, the **Assign access** and **Namespace** values are automatically selected. You can modify those values.

4. From **Import template source** drop-down list, select any of the following values:

   - **GitHub**
   - **GitLab**
   - **Bitbucket server**
   - **From scratch**
   - **From URL (.zip file)**
   - **From folder**
   - **From file(*.zip)**

5. Enter the following details of your selected repository:

   - For **GitLab** or **GitHub**, enter a repository URL, an access token, the relative directory of your service, and the related branch.

     - **GitHub repository URL** - `https://github.com/geekbass/terraform`

       Note: URL parsing is not possible if there exists "/" in branch name as it is not possible to differentiate between sub-directory and branch. You can still use the "/" but must manually update the fields after they are auto-parsed.

     - **GitHub access token**- It is the token required to access GitHub template files. You can find it in your GitHub user account, and is not required for public projects.

     - **GitHub repository download directory**- The path relative to GitHub project location where Terraform template files for this template can be referenced. `vsphere`.

     - **GitHub repository template run directory**- Optional field, the path relative to the GitHub download folder. The Terraform template runs from this folder.

     - **GitHub reference (branch or tag)**- The default value for branch or tag reference is `master`, however, you can change the value.

   - For **Bitbucket server**, enter the following details:

     - **Bitbucket repository URL** - The location where Bitbucket template files can be referenced from.

       Note: URL parsing is not possible if there exists "/" in branch name as it is not possible to differentiate between sub-directory and branch. You can still use the "/" but must manually update the fields after they are auto-parsed.

     - **Bitbucket access token** - The token used to access the Bitbucket template files

     - **Bitbucket repository download directory** - Optional field, the path relative to the Bitbucket server download folder. The Terraform template runs from this folder.

     - **Bitbucket repository template run directory**- Optional field, the path relative to the Bitbucket server download folder. The Terraform template runs from this folder.

     - **Bitbucket reference** - Select **Branch** or **Tag** option and enter the name of the tag or branch.

   - For **From scratch**, enter the following details:

     - **Name** - The name of the template.
     - **Description** - The description of the template.
     - **Cloud providers** - Select the cloud provider for the template.

     Note:

     - The template name must have a combination of only these characters: [a-zA-Z][a-zA-Z0-9_-]
     - For information about the available cloud providers, see Managing connections. If your cloud provider is not in the drop-down list, then select **Other** option. The created template is added to **My templates** in the library page.
     - The template can be associated with more than one provider.

   - For **From URL (.zip file)**, enter the following details:

     - **Name** - Optionally, enter the name of the template. This name overwrites the template name that is found in the repository.
     - **URL** - The location of the zip file (http/https) where the compressed template files can be referenced from. For example, "http://myserver.com/templates/template1.zip".
     - **Template run directory** - Optional field, the sub-folder of the zip file. The Terraform template runs from this folder.
     - **Artifactory API Key** - Optionally, if your .zip file URL is an artifactory location, then enter the artifactory API key.

   - For **From Folder**, enter the following details:

     - **Name** - The name of the template.
     - **Choose folder** - The directory containing the template source code.
     - **Template run directory** - Optional field, the sub-folder of the uploaded folder. The Terraform template runs from this folder.

   - For **From file(*.zip)**, enter the following details:

     - **Name** - The name of the template.

- **Choose file(*.zip)** - The zip archive containing the template source code.
- **Template run directory** - Optional field, the sub-folder of the uploaded folder. The Terraform template runs from this folder.

6. Click **Import**.
7. In the **Template metadata** tab, you can enter or change the template name, icon, short description, cloud provider, and details as a long description or the list of features.
8. In the **Manage template** tab, you can manage the template versions and the template parameters.

   If you are creating a template from scratch, in the **Edit version** tab, add your template source code by importing it from a file or from a URL. In the **Edit version** tab, click **Update parameters** to add parameters from the template source or from a file. For information about using Managed services Variables, see Template Middleware Parameters.

   For information about running a local executable after a resource is created during the deployment, see local-exec provisioner.
9. You can add new version of the template or delete an existing one by clicking the **Manage versions** tab. You can also edit an existing version, disable it, or set it as default by clicking the appropriate action in the overflow menu of the **ACTIONS** column.

   The template version follows `major.minor.patch` convention, where:

   - `major` refers to significant changes
   - `minor` refers to new features added but existing functionality remains broadly compatible
   - `patch` refers to bug fixes

   Note:

   - If you want to define the name of the stack instance in your Terraform template, you do not need to define a Managed services variables or a parameter but you can use the input variable named `ibm_stack_name` that is filled out automatically during the deployment with the name of the stack instance.
   - The character period (.) is not supported in the map name of the template parameters and shared parameters. Example of a map name with a period (.) that is not supported:

     ```
     {
      "my.key1": "myhostname"
     }
     ```

     It causes an internal server error. However, you can use the period (.) in the map value. Example of a map value with a period (.) that is supported:

     ```
     {
      "mykey": "myhostname.com"
     }
     ```
10. Click **Save** to save your template in the library.

    Note: If you created templates with a previous version of Managed services, be aware that the resource names for the IBM Cloud provider were changed. For more information, see Migrating to v0.4.

Reference links

From the overflow menu of a template, you can perform the following actions:

- Deploying a template
- Duplicating a template
- Deleting a template

Editing a template

You can search for a template from the library page.

1. In the navigation bar, click **Library** > **Terraform templates**. The template library is displayed.
2. In the template library page, you can use the following filters to search for a template:

   - Enter the name of the template in the **Search templates** filter.
   - Select the **Cloud provider** to view the list of all of the templates under a particular provider.

     The templates are categorized according to the provider type and template type. The templates you create are available in the **My templates** category. If you select **Other** option when you create the template or the template does not fall under any of the mentioned categories, then you can find your template under **Other** category. The templates that are provided by Managed services

for import functionality are available in the **Import existing** category. The **All templates** includes all the templates irrespective of the categorization.

- Click **Name** or **Created** timestamp header to sort the template records.
- Use show filter icon next to the search text to filter records based on the cloud provider or namespace. The namespace option has **Show Globally accessible** option, which refers to those templates that are globally accessible across all namespaces and are not assigned to any specific namespace.
- You can select a namespace from the **All namespaces** drop-down list. The list of templates in the page changes based on the selected namespace. You can also select "All namespaces" to see all records for which you have access.
- Use pagination to decide the number of records that must fit in a page. Use the **>** or **<** to go through the next or previous set of records.

For steps to import a template from Git repository, see Importing a template.

3. Select **Edit** in the template overflow menu. Alternatively, you can click **View** option in the overflow menu of a template or click the template record to open it. In the **Overview** page, click **Edit**.

   Note: From the template **Overview** page, you can **Duplicate** or **Delete** a template.

4. In the **Template metadata** tab, you do the following tasks:

   - Change the name, short and long description of the template.
   - Add, edit, or delete a feature.
   - Change the icon of the template.
   - Change the **Cloud provider**.
   - You cannot change the value of the namespace. However, you can create a duplicate of this template and associate it with a new namespace. To create a duplicate from this page, you can click the **duplicate** link in the **Assign access** section. Alternatively, you can also select **Duplicate** from the overflow menu. For more information about the procedure to duplicate a template, see Duplicating a template.

5. In the **Manage template** tab, you can edit the following details:

   - In the **Edit version** tab, you can edit the following details:
     - Select the version from the **Version** drop-down list or click **View source** of a version to view the source in read-only mode. If you had used the artifactory repository to import a compressed template file, then you can view the default version and the URL that was used to import the template and also the parameters if present.
     - Click **Update parameters** to update the parameters. In the **Update Parameters** page, you can import from a file or copy paste the details in the editor. Additionally, you can import the parameters from the original **Template source**. You also have an option to edit the parameters by clicking each parameter individually if the source file does not have the `camvariable.json` defined already.
   - In the **Manage versions** page, you can perform the following activities:
     - You can view a table with all version details in it.
     - Click **Add version** to add a new version of this template. In the **Add version** page, enter the following details:
       - Enter version number in the **Display name**.
       - Enter a branch or tag value in **Tag/Branch name**.
       - If you want to include a compressed file, enter the URL pointing to ".zip" file to add a new version
     - From the **Actions** over flow menu of an existing version, you can edit, disable, set as default, or delete a version.
       - If you select the edit option, you can modify the version display name but not the URL.

     Note: The template version follows `major.minor.patch` convention, where:

     - `patch` refers to bug fixes
     - `major` refers to significant changes
     - `minor` refers to new features added but existing functionality remains broadly compatible

     When you upgrade to a new instance through plan/apply for major changes, a compatibility warning is displayed in the plan user interface. For example, when you move from 1.1.0 to 2.1.0, the following warning is displayed:

     "Trying to upgrade to a version that will break compatibility may result in deployment errors. It is recommended to clean up the old instance first and then deploy the new version."

     In this example, version 2 breaks compatibility with version 1, which causes plan/apply to fail. When you upgrade to a new instance for minor or patch changes, then no warning message is displayed because backward compatibility is preserved. For example:

     ```
     - 1.0 to 1.1 or 3.0
     - 1.1.0 to 1.2.0 or 1.1.1
     ```

You can ignore this warning and still proceed with the upgrade. The warning is displayed only if the version starts with a number followed by dot.

6. Click **Save** to save the changes you made to the template.
7. From the overflow menu of the template, you can duplicate, deploy, and delete a template.

Duplicating a template

Procedure to duplicate Managed services templates.

About the task

- If you duplicate a template in Git repository, then changes are not done to the Git repository. However, a duplicate of the record is created in Managed services, which points to the same Git repository as the source template. As a result, both the templates point to the same Git repository and a change to the Git repository impacts both the templates.
- The duplicate template includes copies of all the versions of the source template.

Procedure

1. In the navigation bar, click **Library** > **Terraform templates**. The template library is displayed.
2. In the overflow menu of the template you want to clone, click **Duplicate**. Alternatively, you can create a duplicate from the edit page and overview page.
3. In the **Duplicate template** page, select either of the following options in **Assign access** section:
   - **Make template Globally Accessible** - Templates created globally are accessible by all users.
   - **Make template part of a namespace** - Associate the template with a specific namespace. For this option, you must select a **Namespace** from the drop-down list to which the template must be assigned.

   Note: If you create a template in a specific namespace, then only deployments to that namespace can use the template.
4. Enter the name of the new template.

   Note: Ensure that the name of the template is unique in that namespace for the template. The default name prepopulated is **`Copy of <name of the original template>`**.
5. Click **Duplicate**.
6. In the **Template metadata** and **Managed template** tabs of the newly created template, enter or modify the details. For more information to update the tabs, see Editing a template.
7. Click **Save**. The newly created template is listed in the **My templates** section of the Template Library.

Deleting a template

You can delete an existing template from the template library.

Procedure

1. Go to the Template Library page.
2. From the overflow menu of a template record, select **Delete**.
3. In **Delete template** page, enter "delete" and click **Delete**.

Viewing a template

You can view the details of a template from the library page.

Procedure

1. Open the **Library** > **Terraform templates** page.
2. Search for your template record to view.
3. Click the **View** option from the **Actions** menu of the template record. The details of the template are displayed in **Overview**, **Template source**, and **Parameters** tabs.
4. You can perform the following actions in view mode:
   - Click **Edit** to update the template.
   - Click **Deploy** to deploy the template.
   - From the overflow menu, you can create a **Duplicate** or **Delete** the template.

Changing preloaded content tokens

Starter pack templates are imported from Github and Gitlab. Managed services ships valid tokens to work with these templates. However, you can change to your own Github tokens from the **Template Source** tab of template details page.

Procedure

1. Go to **Library** > **Terraform templates**.
2. Click to open your starter pack template.
3. Open the **Template source** tab.
4. Click the edit icon in **GitHub access token**. Though you cannot see the token, you can delete and update a new token.
5. In the **Edit token** window, enter the new token value and click **Update**.

Creating and managing template instances

Creating and managing template instances and its resources.

- Deploying a template
- Managed services Kubernetes resource controller for Managed services templates
- Viewing template instance details
- Deleting a template instance
- Modifying a template instance
- Managing virtual machine resources
- Importing existing VMware vSphere virtual machine
- Importing an existing PowerVC or OpenStack virtual machine
- Migrating virtual machines using VMware VMotion
- Creating and managing snapshots

Deploying a template

Deploy a template in your Managed services environment.

Before you begin

- VMware templates need to have vmware-tools installed or else the operating system configuration post the deployment is not possible.

Procedure

To deploy a template, follow these steps:

1. In the navigation bar, click **Automate Infrastructure** > **Library**. The library is displayed. Click the **Terraform templates** tab.
2. Click the template that you want to deploy.
3. Read the information about the template and click **Deploy**.

   You can see the template source code in the **Template source** tab and the template parameters in the **Parameters** tab.

4. Specify the following details to deploy a template:

   - **Namespace** - Select the namespace from the drop-down list.

     Note: This step is applicable only for globally accessible templates. You cannot change the namespace during deployment for templates that are associated with a namespace.

   - **Instance name** - Enter a unique name for the instance. The instance name must begin with an alphabet and must have a combination of only these characters: [a-zA-Z][a-zA-Z0-9_-]

   - **Cloud Connection** - Select the cloud connection from the drop-down list. It is a connection to the cloud provider where you want to deploy the template. The cloud connection list only contains the connections defined for the type of cloud provider where you can deploy the template. For information about cloud connections, see Managing connections.

   - **Terraform version** - Select the Terraform version from the drop-down list. For information about creating a user defined Terraform version, see Managing Terraform Versions.

   - If you have used shared parameters while defining your template, then in the **Select data objects**, select the value of all Data Objects associated with the defined Data Types. For the following scenarios, a message is displayed with appropriate information:

     - Your template has an invalid Data Type. A link is provided in the message to edit the template.
     - You do not have any data objects associated with the specified data type. A link is provided in the message to create data objects.

   - If you are deploying enterprise middleware templates from the Enterprise Middleware Repository, select a value for **Advanced content runtime Chef**. This **Advanced content runtime Chef** value is a Data Object that is associated to "advanced_content_runtime_chef" Data Type. If you want your advanced content runtime to be available to middleware template deployments that are in a different namespaces, then modify the data object to be globally available by using the following steps:

     - In the navigation bar, click **Automate Infrastructure** > **Manage**. Click the **Shared parameters** tab.
     - Open the Data Object in edit mode.
     - Select the **Make data object globally accessible (available to all users)** radio button.

     Note: For middleware templates, you can select an Automated Content Runtime that is not deployed on the same cloud provided proper network connectivity exists.

   - Enter all the required deployment parameters. It varies from one template to another based on the definition.

     Note: If you must enter the user ID, then ensure that it corresponds to your operating system and cloud connection. Otherwise, the deployment fails with an error message. For more details, see the General issues section of Troubleshooting deployment issues.

5. Click **Deploy**. The template instance window is displayed. The `Ready` state indicates successful deployment.

   In the template instance window, you can see the instance status, the instance information, the related resource details, and the related input and output variables. You can view or copy the instance logs and you can also access the template from which the instance was deployed.

   Note: Managed services stores the Terraform state file in its database in an encrypted format.

Managed services Kubernetes resource controller for Managed services templates

The Managed services Kubernetes custom resource definition and controller provides you with the ability to deploy, update (Plan/Apply), and delete Managed services templates instances that use Kubernetes yaml resource files.

The templates in Managed services are declarative Terraform templates that allow you to deploy infrastructure as code. With this capability, a single Kubernetes yaml can be created to deploy a composite application that consists of both Kubernetes deployable resources and traditional infrastructure resources, which are deployed with Terraform in Managed services. A typical containerized application is an example of a Kubernetes deployable resource. Examples of traditional infrastructure resources are virtual machines, storage, and so on.

About the task

The Managed services Kubernetes resource controller deploys the Managed services template and waits for SUCCESS status. If the deployment FAILS, then the controller continues to issue PLAN/APPLY requests until the APPLY request returns with SUCCESS status.

If one or more of the template parameter values are changed, then the controller issues a PLAN/APPLY until the APPLY request returns with a status of SUCCESS.

Before you begin

A role binding must be defined for a user or group to have the authority to create/apply a **templates.cloudautomationmgr.ibm.com** resource or to deploy a template in Managed services.

The following is an example for the **icp:camgroup:operator** user group:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
   name: icp:camgroup:operator:template
roleRef:
   apiGroup: rbac.authorization.k8s.io
   kind: ClusterRole
   name: cam-controller-manager-role
subjects:
- apiGroup: rbac.authorization.k8s.io
   kind: Group
   name: icp:camgroup:operator
```

Procedure

- Deploying a Managed services template
- Displaying the status of a Managed services template
- Deleting a deployed Managed services template

Deploying a Managed services template

1. Go to your Kubernetes environment that is same as Managed services.
2. Create your YAML file. The file allows the definition of the template deployment to be placed under the source control that allows changes to be tracked. An example Managed services Kubernetes resource controller yaml file:

   ```
   apiVersion: cloudautomationmgr.ibm.com/v1alpha1
   kind: Template
   metadata:
    name: template-ibmcloud-sample # The name of the resource. This name is used for the Managed services
   deployed instance name. Therefore, the kubernetes resource name and the Managed services deployed
   instance name match with each other.
    namespace: "default"  # Specifies the Kubernetes and Managed services namespace to use for the
   deployment.
   spec:
    connection: "IBMCloud"  # It specifies the name of the Managed services cloud connection to use.
    template:
      name: "Virtual Server with SSH key"  # The name of the Managed services template to deploy.
      version: "2.3"  # The version of the Managed services template to deploy
      parameters:
        parameters:
          datacenter: "dal09"
          hostname: "lrhibmcloud1"
          domain: "raleigh.ibm.com"
   ```

   - **parameters** - It is the collection of name/value pairs that map to the Managed services Template Input Parameter Key and Value.

     ```
     parameters:
       datacenter: "dal09"
       hostname: "lrh1hostname"
     ```

     The following **sharedParameters** parameter is an optional parameter and is dependent on the Managed services template that is deployed:

   - **sharedParameters** - List of Managed services shared parameter data objects to use with the deployed template. For example:

     ```
     sharedParameters:
     - name: DefaultNoBastionHostRequired
       dataType: bastionhost
     ```

   - **secretParameters** - (Optional) List of Managed services secret parameter data objects to use with the deployed template. For example:

     ```
     secretParameters:
     name: vm_password_lrh
     secretKeyRef:
             name: vm-password-lrh-secret
             key: password
     ```

     Where:

     - **name** is the name of the template parameter.

- **secretKeyRef.name** is the name of the Kubernetes secret.
- **secretKeyRef.key** is the name of the secret data defined in the secret.

3. Run the following kubectl command to deploy a Managed services template:

```
kubectl create -f <yaml file name>
```

Note: A template that is deployed by using the Kubernetes controller can be viewed in the deployed template instances page of Managed services user interface.

After deployment, if you want to modify and do a plan/apply, run the following kubectl command:

```
kubectl apply -f <yaml file name>
```

Displaying the status of a Managed services template

1. Verify the status of a deployed Managed services template by using either of the following commands:

```
kubectl describe -f <yaml file name>
```

or

```
kubectl describe templates.cloudautomationmgr.ibm.com nodejs-ibmcloud-test1 -n default
```

Example:

```
Name:         nodejs-ibmcloud-test1
Namespace:    default
Labels:       controller-tools.k8s.io=1.0
API Version:  cloudautomationmgr.ibm.com/v1alpha1
Kind:         Template
Metadata:
  Creation Timestamp:  2019-05-29T20:12:06Z
  Finalizers:
    template.cloudautomationmgr.ibm.com
  Generation:        2
  Resource Version:  7528303
  Self Link:         /apis/cloudautomationmgr.ibm.com/v1alpha1/namespaces/default/templates/nodejs-
ibmcloud-test1
  UID:               08075b85-824e-11e9-b7be-00163e01cbab
Spec:
  Connection:  IBMCloud
  Template:
    Name:  Virtual Server with SSH key
    Parameters:
      Parameters:
        Datacenter:      dal09
        Domain:          raleigh.ibm.com
        Hostname:        lrh1hostname
        Public Ssh Key:  ssh-rsa
    Version:             2.3
Status:
  Action:  CREATE
  Conditions:
    Last Transition Time:  2019-05-29T20:15:41Z
    Reason:                SUCCESS
    Status:                True
    Type:                  CREATE
  Message:
  Outputs:
    Name:    vm_ip
    Type:    string
    Value:   "Public : 9.9.9.9"
  Stack ID:  5ceee797cdb119001c082077
  Status:    SUCCESS
Events:
  Type    Reason              Age    From                     Message
  ----    ------              ----   ----                     -------
  Normal  CREATE:IN_PROGRESS  5m34s  cam-template-controller  CREATE status changed for default/nodejs-
ibmcloud-test1
  Normal  CREATE:SUCCESS      2m1s   cam-template-controller  CREATE status changed for default/nodejs-
ibmcloud-test1
```

The Status fields returned from the **kubectl describe** command:

- Action - Last action performed, that is, CREATE, PLAN, APPLY.
- Message - Last error message, blank if no error message.
- Stack ID - Managed services stack ID of the deployment.

- Status - Status of the last action performed.
- Conditions
  - Last Transition Time - Last update time of the condition.
  - Reason - The status of the Managed services action.
  - Status - It is 'True' when the condition is valid.
  - Type - The type of the condition - the Managed services action (such as CREATE, PLAN, APPLY).
- Outputs
  - Name - Name of the output parameter.
  - Type - The type of the parameter, that is, string, list, map.
  - Value - The value of the output parameter in json format.
- The Event fields returned from the **`kubectl describe`** command are Type, Reason, Age, and From. Example:

```
Type    Reason                 Age   From                        Message
----    ------                 ---   ----                        -------
Normal  CREATE:IN_PROGRESS     15m   cam-template-controller     CREATE status changed for
default/nodejs-ibmcloud-1
Normal  CREATE:SUCCESS         69s   cam-template-controller     CREATE status changed for
default/nodejs-ibmcloud-1
```

2. Optionally, you can also use Managed services Kubernetes controller startup parameters - controller parameter. They are defined in **`cam-controller-statefulset.yaml`** file. The args parameters are specified in the pod definition.

   The **`-syncPeriod`** is a controller parameter for the time duration between resource status checks. Value units are 's', 'm', 'h'." (Default is 2m).

   For example:

   - `-syncPeriod=2h30m`
   - `-syncPeriod=300s`
   - `-syncPeriod=30m`

Deleting a deployed Managed services template

If you no longer require the deployed instance of a template, run either of the following commands to delete it:

```
kubectl delete -f <yaml file name>
```

or

```
kubectl delete templates.cloudautomationmgr.ibm.com nodejs-ibmcloud-test1 -n default
```

Note: The command deletes the Managed services resources for the deployed instance and then the Managed services instance. After the Managed services instance is successfully deleted, the Kubernetes resource that represents the Managed services instance is also deleted.

Viewing template instance details

After you deploy a template, you can view the related instance details and perform actions on the related resources.

You can also use the Managed services API to import a virtual machine as a stack instance. The instance details for the imported virtual machine can be viewed and actions performed on the related resources. For more information, see Importing existing VMware vSphere virtual machine or Importing an existing PowerVC or OpenStack virtual machine.

Procedure

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Search for your instance.

   Note: You can also use the following search filters:

   - Enter the name of the instance in the search instance filter.
   - Click the show filter icon to list template instances based on the selected provider.
   - You can click the column headings to sort the list.
   - You can select a namespace from the **All namespaces** drop-down list. The list of template instances in the page changes based on the selected namespace. You can also select "All namespaces" to see all the template instances for which you have access.

   Use the **Instances per page** drop-down to select the number of instances to display per page. Use the **>** or **<** to go to the next or previous page.

3. Click the instance for which you want to see the details. The **Overview** tab of the instance is displayed.

    - From the **Overview** tab, you can see the instance status, template information, instance information, resource details, and the input and output variables. For each resource, you can access the console (if existing) and perform the actions that are listed by clicking the overflow menu.

      Note: To access the KVM console in the IBM Cloud Customer Portal, you must follow the procedure described in Access the KVM Console.

    - From the **Modify** tab, you can modify the instance. For more information about how to modify Resources and Parameters, see Modifying a deployed template instance.

    - From the **Log file** tab, you can view and copy the instance log.

      Note: If the Log exceeds the maximum size of 200 KB, then only the last 200 KB of the Log is displayed.

Deleting a template instance

You can delete a template instance and all its related resources when you do not need them any more.

About the task

In Managed services, you can create templates based on IBM PureApplication patterns. Delete template instance is also supported for such templates.

Procedure

To delete a template instance, follow these steps:

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Search for your instance.

   You can use the following methods to search for template instances:

   - Enter the name of the instance in the search instance filter.
   - Click the show filter menu icon to list template instances based on the selected provider.
   - You can click the column headings to sort the list.
   - You can select a namespace from the **All namespaces** drop-down list. The list of template instances in the page changes based on the selected namespace. You can also select "All namespaces" to see all the template instances for which you have access.

   Use the **Instances per page** drop-down to select the number of instances to display per page. Use the **>** or **<** to go to the next or previous page.
3. In the instance list, click the overflow menu for the instance that you want to delete. The action list is displayed.
4. In the action list, click **Destroy resources** to delete all the resources related to the instance from the cloud provider. This option does not delete the instance from the Managed services environment.
5. Enter **`destroy`** to confirm and click **Destroy**. The instance status is changed to **`Destroyed`**.
6. In the action list, click **Delete instance** to delete the instance from the Managed services environment.
7. Enter **`delete`** to confirm and click **Delete**. The instance is deleted from the instances list.

   Note:

   - If you delete a template, then you cannot modify or deploy instances that are associated with this template.
   - If you are using AWS cloud connections with an expired temporary access key, then the destroy operation fails. Refresh the credentials in the cloud connections page and run a plan and apply before you retry the destroy a template instance operation. For steps to refresh the credentials, see Configuring an Amazon EC2 connection.

Modifying a template instance

You can redeploy an existing instance after selecting the desired template version and modifying the input parameters.

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.

2. Search for your instance.

   Note: You can also use the following search filters:

   - Enter the name of the instance in the search instance filter.
   - Click the show filter icon to list template instances based on the selected provider.
   - You can click the column headings to sort the list.
   - You can select a namespace from the **All namespaces** drop-down list. The list of template instances in the page changes based on the selected namespace. You can select "All namespaces" to see all the template instances for which you have access.

   Use the **Instances per page** drop-down to select the number of instances to display per page. Use the **>** or **<** to go to the next or previous page.

3. Click to open a deployed instance.

4. Optionally, you can manage resources. For more information, see Modifying virtual machine resources.

5. Click **Modify** tab.

6. In the **Modify deployed instance** page, optionally, enter the following details:

   1. You can select a different template from the drop-down list. The list includes all templates that are available for all namespaces. Before you change the template itself, ensure that you are aware of its impact.

      Notes:

      - Even after changing the template, the original template and version details are available in read-only mode for reference.
      - The word `current` is added to the currently deployed template.
      - For the modified instance to apply without any issues, ensure that the newly selected template is compatible with the original template. Otherwise, issues may occur during plan/apply.

   2. Select a new template version from the drop-down list.

      Notes:

      - The word "current" next to a version indicates that it is the version with which the instance was deployed. You can verify the same in the **Instance details** section of the **Overview** tab.
      - A version change can add or remove parameters.
      - If you had selected a different template in the **Choose template** drop-down list, then the default version of that template is populated here.
      - The template version follows `major.minor.patch` convention where:
        - `major` refers to significant changes
        - `minor` refers to new features added but existing functionality remains broadly compatible
        - `patch` refers to bug fixes

7. Click **Next**.

8. In the **Edit parameters** tab, enter the following details:

   1. Select a cloud connection.

   2. **Terraform engine version** - Select the Terraform version from the drop-down list. For information about creating a user defined Terraform version, see Managing Terraform Versions.

   3. If the selected version has Data Types defined, then select values for the associated Data Objects from the drop down list. For the following scenarios, a message is displayed with appropriate information:

      - Your template has a invalid Data Type. A link is provided in the message to edit the template.
      - The specified data object is not associated with the data type or the data object is invalid. A link is provided in the message to create data objects.

      Notes:

      - In the **Overview** tab > **Input parameters** section, you can view the input parameter values used when the instance was deployed.
      - If there are no data objects associated with the datatype, then you will be prompted to either create a data object or manually enter values in this **Edit parameters** page.

   4. Enter values in **Additional options**.

      Notes

      - Whenever you change a parameter value, an undo icon is displayed. Click the icon to undo your changes and revert it to default values.
      - The default value for an the existing parameter is the value that was used during the previous deployment. The value used during the previous deployment takes precedence over the value defined in the template.
      - Consider the following points whenever you select a different version or a new template:

- If there are new parameters in the selected version or template that do not match with the previously deployed template, then they are displayed with the word "new" in parenthesis.
- The default value for the new parameters are the template values.
- If a new parameter is not available in the selected version, then it is not displayed.
- If a Data Object value is injected for a parameter in the template definition of the selected version, then the appropriate output parameter value is used as the default value.
- If parameters are grouped in the selected template version, then they are displayed as separate collapsible sections. The headings of these sections are displayed in the **Edit parameters** tab for easy navigation.
- If `immutable` is set to true for a parameter in the selected version, then you cannot edit the default value of the parameter.
- If `immutable_after_create` is set to true for a parameter in the selected version, then you can enter a value and deploy it for the first time. You cannot edit it in subsequent plan changes.

9. Click **Plan changes**.

   Notes:

   - If you perform a resource taint activity at this stage, then the resource marked as tainted is not considered for the current completed plan.
   - When you taint a virtual machine resource, Managed services marks only that resource and recreates it during the next plan or apply. It does not recreate all resources on the selected virtual machine based on the template logic. If you want to recreate all associated resources of the virtual machine, then you must identify and individually taint all the required resources.

10. In the **Planned changes** page, wait for the status to change from **In progress** to **Success**. Alternatively, go to the **Activity** section of the **Overview** tab to see whether the status of the plan is complete. If the status displays **Failure**, then it indicates that the redeployment with the modified value can result in failure.

    Notes:

    - The resources are neither destroyed or created at this stage, but the **Log output** shows the outcome of your changes beforehand.
    - If the plan is in complete state, then you can revisit the modify page of the deployed instance at a later time and apply the changes.

11. Verify the **Log output** to reconfirm your changes. The log also provides information about what happens if the changes are applied. You can always see the log details for all completed activities in the **Activity** section of the **Overview** tab.

12. Click **Apply changes**. The instance gets redeployed with your latest modifications. At this stage, the resources are created and destroyed as per your plan.

13. In the **Apply changes** window, enter "apply" in **Confirm changes** and click **Apply**. The redeployment starts and you can see the status in the **Overview** tab next to the name of the deployed template instance. When the status changes to **Ready**, it indicates that the redeployment has completed successfully.

    Note:

    - You need to confirm here because the redeployment of your changes may result in adding, modifying, or destroying the associated resources.
    - When you click **Apply**, the template instance moves to the new version irrespective of success or failure. It is because the apply might destroy some of the resources before the occurrence of failure.

14. You can also view the status of the deployment details in the **Activity** section of the **Overview** tab. The activity also shows the log of plan changes. If the template or version is changed in the plan/apply step and changes are applied, then you can see new changes in the **Overview** tab.

Managing virtual machine resources

You can manage resources from the **Overview** tab of the deployed template instances page.

You can also run the import a VMware virtual machine API and manage resources from the deployed template instance page of Managed services. For more information about the API, see Common Managed services APIs.

- Start and stop virtual machine resource
- Reset
- Shutdown OS
- Restart OS
- Snapshots
- Taint and untaint resource
- Details

The following table lists Managed services limitations for Cloud provider:

Cloud Provider	Limitations

Cloud Provider	Limitations
IBM PureApplication	You can create templates based on IBM PureApplication patterns. You can start/stop virtual machines and destroy resources for such templates, but the taint/untaint feature is not supported.
IBM Cloud, Amazon AWS, and VMware vSphere	Start, stop, and shutdown are available for these providers. Reset, and View Snapshots are available for VMware vSphere only. Restart OS option is available for both Amazon AWS and VMware vSphere.
OpenStack, Google Cloud, Microsoft Azure, and IBM PureApplication	Start and Shutdown are available for these providers. **Note**: Stop (forced power off) is not supported for these providers. Instead the provider attempts a graceful operating system shutdown first. If shutdown does not complete in a given amount of time, then the provider forcefully powers off the virtual machine.
Nutanix	You cannot manage the resources for this provider in this release.
VMware NSXv, VMware NSX-T, IBM Cloud Pak for AIOps and IBM Cloud Kubernetes Service	Manage resources section is not applicable for these providers as they do not provision virtual machines.

Start and stop virtual machine resource

The start action powers on a stopped virtual machine and stop action does a force power off of the virtual machine without operating system shutdown.

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Click to open a deployed instance.
3. In the **Overview** tab, go to the **Resource details** section.
4. From the overflow menu of a resource, click **Start**. The **STATE** of the virtual machine resource changes from "Stopped" to "Running". Likewise, click **Stop** to change the **STATE** of the virtual machine resource from "Running" to "Stopped". The start/stop option is available only when the resource is of type `virtual_machine` and is in a powerstate.

Reset virtual machine resource

You can reset a VMware virtual machine resource. It resets and restarts the virtual machine.

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Click to open a deployed instance.
3. In the **Overview** tab, go to the **Resource details** section.
4. From the overflow menu of a resource, click **Reset**.
5. In the **Activity** section of the **Overview** page, you can view the status of the reset activity and also its logs.
6. Optionally, you can verify the status of the virtual machine in VMware.

Shutdown virtual machine resource

The shutdown action performs a graceful operating system shutdown and then powers off the virtual machine.

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Click to open a deployed instance.
3. In the **Overview** tab, go to the **Resource details** section.
4. From the overflow menu of a resource, click **Shutdown OS**. The **STATE** of the virtual machine resource changes from "Running" to "Stopped".

Restart OS

Restart OS action is available for VMware vSphere and Amazon AWS. This option is enabled only when the virtual machine power state is ON. On Amazon AWS, this option corresponds to a reboot action.

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Click to open a deployed instance.
3. In the **Overview** tab, go to the **Resource details** section.
4. From the overflow menu of a resource, click **Restart OS**. The operating system of the virtual machine is restarted.

Snapshots

Snapshots are available only for a VMware virtual machine.

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Click to open a deployed instance.
3. In the **Overview** tab, go to the **Resource details** section.
4. From the overflow menu of a resource, click **Snapshots** to view the details of a snapshot in read-only mode.

Taint and Untaint resource

Taint marks a Terraform managed resource as tainted and the tainted resource is deleted and recreated during the next apply.

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Click to open a deployed instance.
3. In the **Overview** tab, go to the **Resource details** section.
4. From the overflow menu of a resource, click **Taint**.
5. In the **Taint resource** confirmation window, click **Taint**. In the **Other** tab of the **Activity** section, a "Taint resource is in progress" entry is displayed. After the activity completes successfully, the activity changes to "Taint resource succeeded" with a green check mark.
6. You can click to view the log of the activity.
7. For a tainted resource, you can go to the overflow menu of the resource and click **Untaint** before you apply the changes. Untaint unmarks a tainted resource and restores it.
8. After you apply the changes, a resource that is marked as tainted gets destroyed and recreated. The **Taint** option is again available for a recreated resource.
9. You can see the status of the action in the **Activity** section.

Details

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Click to open a deployed instance.
3. In the **Overview** tab, go to the **Resource details** section.
4. From the overflow menu of a resource, click **Details**. All the details of the resource are displayed in the **Instance details** page. As an administrator or cluster administrator, you can hide some details of the resource. For more information about the APIs, see Retrieve all IaaS resource settings and Retrieve a specific IaaS resource settings.

Importing existing VMware vSphere virtual machine

You can import a VMware vSphere virtual machine as a Managed services stack instance, identify whether the instance is imported successfully, and manage the stack resources.

A default provided template, **`Import existing VM with a single disk - VMware vSphere for Terraform 1.x.x`**, is shipped as a sample to import a virtual machine. You can use this template during the import of VMware virtual machines. However, this template cannot be used to deploy new virtual machines.

Procedure

1. Use the **`Import VM by IDs`** or **`Import VM by Name`** to import a VMware vSphere virtual machine as a stack instance. For more information, see Import VM by IDs API and Import VM by Name API.
2. Log in to IBM Cloud Pak for AIOps console.
3. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
4. Click the deployed instance to open it and manage the imported virtual machine resources.
5. From the **Overview** tab > **Resources** section, use the overflow menu to start, stop, shutdown, taint, untaint, and view details. For more information, see Managing virtual machine resources.
6. You can also modify the virtual machine using the plan and apply process. For more information, see Modifying a deployed template instance.

Limitations

- The **`Import existing VM with a single disk - VMware vSphere for Terraform 1.x.x`** template works only for virtual machines with a single disk. Go to the template source to find the following single disk information:

  ```
  disk{
    unit_number = "${var.disk_0_unit_number}"
    path = "${var.disk_0_path}"
    label = "${var.disk_0_label}"
    size = "${var.disk_0_size}"
    datastore_id="${var.disk_0_datastore_id}"
  }
  ```

  If you are importing a virtual machine with more than one disk, clone this template to create a copy and duplicate the disk section for each additional disk. Make sure to increment the disk reference number in the variable names.

- The parameter values of **`Import existing VM with a single disk - VMware vSphere for Terraform 1.x.x`** template are automatically populated after a successful import. When the mapping occurs, if your variable values have dots (.), then Managed services

automatically converts them to underscore (_) and sets the new values for the template variables. The conversion occurs because the dot (.) character is not supported for the template variable values.

Importing an existing PowerVC or OpenStack virtual machine

You can import PowerVC and OpenStack virtual machines as Managed services stack instances, identify whether the instances are imported successfully, and manage the stack resources.

A default provided template, **`Import existing VM - PowerVC and OpenStack for Terraform 1.x.x`**, is shipped as a sample to import a virtual machine. You can use this template to import virtual machines, but it cannot be used to deploy new virtual machines.

Procedure

1. Use the Managed services API to import virtual machine as a stack instance. For more information, see Import VM by IDs API and Import VM by Name API.

   Note:

   - The **`power_state`** property is discovered during import and is set to the power state of the virtual machine when it is imported.
   - The **`force_delete`** and **`stop_before_destroy`** properties are not discovered during import. These properties are given default values based on their default value in **`camvariables.json`**. These are non-destructive properties so they can easily be changed by modifying the instance.
2. Log in to IBM Cloud Pak for AIOps console.
3. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
4. Click the deployed instance to open it and manage the imported virtual machine resources.
5. From the **Overview** tab > **Resources** section, use the overflow menu to start, shutdown, taint, untaint, and view details. For more information, see Managing virtual machine resources.

   Note: If you make changes to the instance, be aware that the current virtual machine **`power_state`** may not match the discovered **`power_state`**. For example, if a virtual machine in "shutoff" state was active during import, then the value of the **`power_state`** property is still "active" when you modify the instance. Change the value to "shutoff", or else the virtual machine gets started after you apply your changes.
6. You can also modify the virtual machine using the plan and apply process. For more information, see Modifying a deployed template instance.

Resizing a PowerVC Virtual Machine

Shut down your PowerVC virtual machine before resizing it. To resize the virtual machine, do the following steps:

1. View the instance details for the imported virtual machine.
2. In the **Resource details** section, click **Shutdown OS** from the overflow menu of the virtual machine resource.
3. Modify the instance after the virtual machine is shutdown.
4. Change the flavor to a desired value.
5. Ensure the Power State is set to "shutoff".
6. Plan and apply your changes.
7. After the changes are applied, click **Restart OS** from the **Actions** overflow menu of the virtual machine resource.

For steps to shutdown and restart OS of the virtual machine, see Managing virtual machine resources.

Limitations

- Storage attached to the imported virtual machine is not discovered during import. This includes volume attachments as well as any block devices specified when the virtual machine got created.
- If the imported virtual machine was created using a bootable volume rather than an image, then the **Image Name** input variable value may be “Attempt to boot from volume – no image supplied”. If you make changes to the instance that results in the virtual machine being recreated, the virtual machine gets destroyed but does not get created. This issue occurs because block devices are not discovered during import.

Migrating virtual machines using VMware vMotion

Managed services supports VMware vMotion to migrate virtual machines between VMware ESX servers and datastores. In addition, it also supports the migration of virtual machines from one vCenter to another. You can perform the migration task from both VMware vSphere console and Managed services user interface.

- Steps to migrate virtual machine from Managed services
- Steps to migrate virtual machine from VMware vSphere client

Note: VMware vMotion is not supported in a cross-cluster scenario.

For more information about VMware vMotion, see VMware Docs.

Steps to migrate virtual machine from Managed services

1. Log in to Managed services.
2. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
3. Click to open a deployed instance.
4. Click **Modify** tab.
5. In the **Modify deployed instance** page, optionally, select a new template version from the drop down list.
6. If the selected version has Data Types defined, then you select values for the associated Data Objects.
7. In the **Edit parameters** tab, change the values for **Cluster** and **Storage**.
8. Plan and apply those changes to destroy the existing virtual machines and redeploy them to the new Host or Storage.
9. Go to Terraform logs to check the success of the migration.
10. To verify the migration in VMware vSphere client console, go to **Recent tasks** and see whether the datastore or the host has changed accordingly.

Steps to migrate virtual machine from VMware vSphere client

From the VMware vSphere client console, migrate a virtual machine through vMotion onto a different VMware ESX server / cluster and datastore.

For more information about VMware Vmotion, see VMware Docs.

Migrate to a different host

1. Log in to VMware vSphere client console.
2. Right-click the virtual machine and select **Migrate**.
3. Go to **Migrate Virtual Machine**.
4. Select the **Change host** as the Migration Type and click **Next**.
5. Select the destination host and click **Next**. A compatibility evaluation between the source and target is done and the status is displayed.
6. Select the vMotion priority and click **Next**. The recommended high priority option reserves CPU for optimal vMotion performance. You can also choose to migrate with available CPU resources.
7. Retain the default settings and click **Finish**.
8. Log in to Managed services and check whether you are able to work with the migrated virtual machine.

   Note: Example tasks to work with the migrated virtual machine are start, stop, plan, apply, and destroy the virtual machine.

Migrate to a different datastore

1. Log in to VMware vSphere client console.
2. Right-click the virtual machine and select **Migrate**.
3. Go to **Migrate Virtual Machine**.
4. Select the **Change datastore** as the Migration Type and click **Next**.
5. Select the destination datastore and click **Next**.
6. Retain the default settings and click **Finish**.
7. Log in to Managed services and check whether you are able work with the migrated virtual machine.

   Note: Example tasks to work with the migrated virtual machine are start, stop, plan, apply, and destroy the virtual machine.

Creating and managing snapshots

You can create and manage virtual machine snapshots for a VMware instance.

Create a snapshot

Steps to create a new snapshot of a virtual machine.

1. In the navigation bar, click **Automate Infrastructure** > **Instances**. Click the **Terraform templates** tab.
2. Open a VMWare deployed instance and click the **Snapshots** tab. Alternatively, in the **Resource Details** section, select **View Snapshot** from the overflow menu. The **Snapshots** tab opens, and a resource is selected by default. The snapshots of that selected resource are also displayed.
3. Click **Create Snapshot** to create a new snapshot.
4. Enter the **Name** and an optional **Description** of the snapshot.
5. If **Snapshot virtual machine's memory (needs more time)** and **Quiesce guest file system (Needs VMware tools installed)** options are in enabled state, select as per your requirement. These two options are enabled only if your virtual machine is in power on state. In case of power off state, the **No additional features** option is selected by default.
6. Click **Create**. A notification is displayed confirming the successful creation of the snapshot. The newly created snapshot is listed along with the other snapshots of the virtual machine. You can view the logs in the **Activity** > **Others** section of the instance **Overview** tab. Click **View Details** to view details of a snapshot log record.

Manage snapshots

You can view details of a snapshot, delete a snapshot, revert to a snapshot, or revert to the latest snapshot.

- From the overflow menu of a snapshot, perform the following operations to manage a snapshot:
    - Click **View Details** to view the details of a snapshot in read-only mode. Click **Close** to go back to the **Snapshots** tab.
    - To revert to a particular snapshot, click **Revert to snapshot**.
    - To delete a snapshot, click **Delete** and enter "delete" in the confirmation window then click **Delete**.
- To restore to the parent snapshot of the current state of a virtual machine, click **Revert to Latest**. Enter "revert" in the confirmation window and click **Revert** to complete the operation.

Note: Managed services adds the size of the following associated files to determine the overall size of a snapshot:

- .mem - File created if the memory flag is set to "true". It stores the memory state of the virtual machine.
- .vmsn - Metadata file for a snapshot.
- .vmdk - Files created to store the difference of the current state of the disk with the state of the disk at the time of snapshot creation.

These files are associated with a specific snapshot. However, the .vmdk files of the parent snapshots are not included.

Managed services

By using the Managed services, you can automate provisioning of infrastructure and virtual machine applications across multiple cloud environments with optional workflow orchestration.

The Managed services was previously called as Terraform & Service Automation or IBM Cloud Automation Manager. For documentation for previous versions of IBM Cloud Automation Manager, see the IBM Cloud Automation Manager Documentation.

Managed services is a multi-cloud, self-service management platform running on IBM Cloud Pak® for AIOps that empowers developers and administrators to meet business demands.

Service Composer from the Managed services allows you to expose cloud services to any cloud through the Service library. It accelerates developer velocity and improves governance.

Managed services uses open source Terraform to manage and deliver cloud infrastructure as code. Cloud infrastructure delivered as code is reusable, able to be placed under version control, shared across distributed teams, and it can be used to easily replicate environments. Managed services brings a common approach to automate all workloads in all clouds to reduce configuration errors and improve operational efficiency. You can manage infrastructure as code using HashiCorp Terraform software.

The Managed services content library comes pre-populated with sample templates to help you get started quickly. It allows you to leverage IBM, community, and home-grown assets as building blocks and modernize legacy applications and build new cloud native applications. Use the sample templates as is or customize them as needed. A Chef runtime environment can also be deployed using Managed services for more advanced application configuration and deployment.

With Managed services, you can provision cloud infrastructure and accelerate application delivery into multiple cloud environments with a single user experience.

You can spend more time building applications and less time building environments when cloud infrastructure is delivered with automation. You are able to get started fast with pre-built infrastructure from the Managed services library.

To access the Managed services connection and configuration section, log in to IBM Cloud Pak® for AIOps console and click **Automate infrastructure>Manage**.

To access the Managed services template and Service library, log in to IBM Cloud Pak® for AIOps console and click **Automate infrastructure>Service library**.

To access the Managed services template and Service instances, log in to IBM Cloud Pak® for AIOps console and click **Automate infrastructure>Service instance**.

To access the Service catalog, log in to IBM Cloud Pak® for AIOps console and click **Automate infrastructure>Service catalog**.

To deploy infrastructure, you can orchestrate a flow with a wide variety of modules and integrations.

- Creating and managing services
- Creating and managing service instances
- Creating and managing custom actions
- Managed services Content

Creating and managing services

Managed services provide end to end IT management services whose deployment is similar to any other database server deployment on a cloud environment. You can create, edit, order, and maintain services from the Manage service Library page. Services include one or many components, such as user interaction, template, custom actions, and orchestration logic.

For troubleshooting and limitations in the service composer, see Troubleshooting Service Composer or Portal issues.

Tasks that you can perform to manage services and versions are as follows:

- Service level actions
- Service version level actions
- Sample services are provided by Managed services in the Manage service library of GIT repository. For the entire list of services, see Managing services in service library.
- Search for a service:
  - In the Manage service library page, the services are sorted according to their category. You can sort your view of the services based on service version statuses and you can toggle which status filter is available.
  - You can use the **Search services** filter. The service tile has the service icon, name, status, and overflow menu.
- Create a service instance - You can create service instances, customize them based on your requirement, and use it in the overall application development.
  - Create a service instance from Managed services
- View service instance details
- Terminate a service instance
- Create bind instances for service instances
- Resume or retry a service instance

Note: Each service can have multiple versions. If you want to use a single version service, you need to set the COMPATIBILITY_MODE to TRUE. To set this mode:

1. Make sure you have Kubernetes CLI installed.
2. Run the following command:

```
kubectl set env deployment/cam-service-composer-api COMPATIBILITY_MODE=true -n cp4aiops
```

You can now use a single version service.

Inbuilt services in Managed services

The GitHub repository includes services in JSON format. Many of these services are inbuilt in Managed services automatically. For services that are not automatically imported, you must manually download, import, and deploy them in Managed services.

You can customize or delete these inbuilt services from your Managed services user interface. To synchronize the Managed services Service Library page with GitHub repository, start or restart the `composer-api` pod. For reference, see GitHub repository link.

In Managed services user interface, go to **Library** > **Services** and view them in the following categories:

- Cluster Lifecycle Services
- Cloud Services
- Hybrid Cloud
- Approval
- Virtual machine

Note: All the services that are listed here are globally accessible from Managed services module. But to access these services in the Manage service Library of IBM Cloud Pak for AIOps, you need to duplicate these services in the user assigned namespace and then publish it from Managed services. For more information, see Duplicating a service and Publishing a service.

Cluster Lifecycle Services

These services create cloud provider-specific Kubernetes cluster and OpenShift clusters, and import them to Infrastructure management installed with Red Hat® Advanced Cluster Management for Kubernetes or without Red Hat Advanced Cluster Management.

Inbuilt services within the Cluster Lifecycle Services category to create a cluster and import them to Red Hat Advanced Cluster Management

Service name with GitHub link	Description
Google Kubernetes Engine Cluster with Red Hat Advanced Cluster Management import	Deploys a Google Kubernetes Engine cluster and imports the cluster into a Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.
Amazon Elastic Kubernetes Service Cluster with Red Hat Advanced Cluster Management import	Deploys an Amazon Elastic Cloud Kubernetes Service cluster and imports the cluster into a Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.
Azure Kubernetes Service Cluster with Red Hat Advanced Cluster Management import	Deploys a Microsoft Azure Kubernetes Service cluster and imports the cluster into a Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.
Red Hat OpenShift Kubernetes Service on IBM Cloud with Red Hat Advanced Cluster Management import	Deploys a Red Hat OpenShift Kubernetes Service cluster on IBM Cloud and imports the cluster into a Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.

Cloud Services

The following table lists all inbuilt services that can connect to Public Cloud Services

Service name with GitHub link	Description
Amazon Relational Database Service	This service configuration uses the AWS provider to provision a Relational Database Service (RDS) on AWS.
Amazon S3 Bucket	This service configuration uses the AWS provider to provision a S3 bucket on AWS.
IBM Database Service	This service configuration uses the IBM Terraform provider to provision a Database Service IBM Cloud.

Hybrid Cloud and ServiceNow CMDB Integration

The following table lists all inbuilt services within the Hybrid Cloud category:

Service name with GitHub link	Description
Multi Cloud - MEAN Stack deployment	Deploys a hybrid cloud MEAN stack (Mongo, Express, Angular, Node) on your choice of cloud. This deployment also deploys a simple sample application to validate the stack is operational. After the stack gets deployed and CMDB gets updated with server records.

Approval

The following table lists the inbuilt services that showcase integration with Infrastructure Management approval process.

Service name with GitHub link	Description
Infrastructure Management Approval	This service is used to get approval from Infrastructure Management before provisioning the resources.

Virtual machine

The following table lists all inbuilt services within the virtual machine. These services can be used to stand up a virtual machine in clouds such as IBM Cloud, Amazon Web Services, Google Cloud Platform, Azure, and VMware.

Service name with GitHub link	Description
Virtual machine on Amazon Web Services	Creates a virtual machine instance on Amazon Web Services
Virtual machine on Google Cloud Platform	Creates a virtual machine instance on Google Cloud Platform
Virtual machine on IBM Cloud	Creates a virtual machine instance on IBM Cloud
Virtual machine on Microsoft Azure	Creates a virtual machine instance on Microsoft Azure
Virtual machine on VMware	Creates a virtual machine instance on VMware

Managing categories

Create and manage categories to organize your services in the library.

- Creating categories
- Deleting a category

Creating categories

Create a category to organize your services in the library.

About the task

The "Cluster Lifecycle Services" category is preloaded for you and you cannot delete it.

Procedure

To create a category, follow these steps:

1. In the navigation bar, click **Automate Infrastructure** > **Library**. Click the **Services** tab.
2. In the **Service categories** pane, click **Add category**.
3. Enter a name for the category and click **Add**. When the category is created, a confirmation message is displayed.

Alternatively, when you create a service, select the **Add a category** item in the **Category** drop-down to create a new category. The naming convention for category name is as follows:

- Must start with a letter
- May contain alphanumeric, underscore, hyphen, and space characters

Deleting a category

You can delete a category from the library.

About the task

The "Cluster Lifecycle Services" category is preloaded for you and you cannot delete it.

Procedure

To remove an empty category, follow these steps:

1. In the navigation bar, click **Automate Infrastructure** > **Library**. Click the **Services** tab.
2. Hover over the category you wish to delete and move your cursor to the right until the minus icon appears. Click the minus icon. A message is displayed asking you to confirm the remove operation. If any services use the category, then the remove operation will fail. Move all services in this category to another category and then remove it.
3. Click **Remove** in the confirmation message. After the category is successfully removed from the library, a message is displayed.
4. Click **Close**.

Bookmarking Service Library URL with preselected category

This feature is included to navigate to a specific category in the Manage service library page.

Procedure

1. In the navigation bar, click **Automate Infrastructure** > **Library**. Click the **Services** tab.
2. Click a category in the **Services** pane. For example, click the "Cluster Lifecycle Services" category.

   The selected category name is appended to Managed services user interface URL.
3. Copy that URL.

   For example, `https://<TSA_HOSTNAME>/cam/portal/#!/servicelibrary/category/Cluster%20Lifecycle%20Services`. Where, `Cluster Lifecycle Services` is the name of the selected category.
4. Save or share the URL for future use.
5. Use that URL to navigate to the specific category in the Manage service library.

Creating a service

You can create or import a service from the Managed services service library page.

About the task

If there are any problems while you create, save, or publish your service, then warning or error messages are displayed. For more information on these service validations, see Service validations.

Procedure

1. In the navigation bar, click **Automate Infrastructure** > **Library**. Click the **Services** tab.
2. Click **Create service**.
3. In the **Create service** window, enter the following details:
   - Select either of the following options in **Assign access** section:
     - **Make this service globally accessible (available for all users)** - Services created globally are accessible to all users.
     - **Make this service part of a namespace** - Associate the service with a specific namespace. For this option, you must select a namespace from the drop-down list to which the service will be assigned.

     Notes:
     - Based on the value selected in the namespace drop-down list of the **Library** > **Services** page, the **Assign access** and **Namespace** values are automatically selected. Modify the values if you want.
     - If the service is defined as globally accessible, then only globally accessible resources are available in the composer. For example, globally accessible cloud connections, shared parameters, and content runtimes are only available in the service composition tab for a service that is defined as globally accessible.
     - If the service is defined in a namespace context, then only globally accessible resources and resources defined in the same namespace context are available in the composition tab of the service.
   - Enter the service name.
   - Enter the version name. This version is used as a default version if not specified. The version format must follow **V.R.M.F** where the letters are integers in range 0-99.
   - Optionally, select a category.

     If you do not specify any category, then the service is assigned to the **Cloud Services** category. You can select **Add a category** from the **Category** drop-down list to create a new category. If you select **Add a category**, then enter the name of the new category and click **Create**. The **Select from list** allows you to go back to the original list of categories.

     For information about categories, see Creating a category.

Notes:

- You can create a service with the same name in a different namespace, but it is not allowed within the same namespace.
- Service name must follow the Kubernetes naming convention, or else the published services do not get listed in IBM Cloud Pak for AIOps catalog:
  - Alphanumeric, underscore, hyphen, and space
  - Starts with a letter
- If you plan to publish a service, ensure that its name is less than 64 characters long. If you have more than 64 characters, then publishing the service to IBM Cloud Pak for AIOps may cause issues.

4. Click **Create**. The service details page is displayed. It includes **Overview**, **Composition**, **Parameters**, **Plans and form**, and **Source code** tabs.

5. In the service details page, a drop-down list of all service versions is displayed next to the service name. Select the version to view service details for that service version.

6. In the **Overview** tab, enter **Service name**, **Short description**, and **Long description** in the **Quick overview** section.

7. In the **Select icon** section, you can select or change the existing icon for the service.

8. In the **Version details** section, you can View or Edit the version details from the overflow menu. You can also set a particular version as default by clicking **Set default version** from the overflow menu. By default, the version created during the **Create service** modal is the default service version.

9. In the **Change version** drop-down list, you can change the version and view/load the details of that service version.

10. Click **Create version** to create a new version based on any of the existing versions. See Adding a Service Version. The created version is added to the drop-down list displayed next to the service name.

11. In the **Tags** section, select **System tags** to include all predefined tags. You cannot delete system tags. In edit mode, you can only change value of **Name**.

    Notes:

    - The following parameters are available as system tags:
      - `environment` - The environment in which this service is running.
      - `Name` - Name you provide for the service instance at the time of request.
      - `request_group` - The current context group id of the current user that requested the provisioning of the template. This is important because a user may be part of more than one group and this identifies the group for which the user made the request.
      - `request_user` - The user id of the current user that requested the provisioning of the template.
      - `service_identifier` - Unique identifier generated by Managed services, which is mapped to the service instance of the template provisioned.
      - `service_name` - Name you provide for the service instance at the time of request.
    - Managed services converts the following non-permissible characters in the system tag definition to permissible characters:
      - Special characters - They are converted to "_". For example, if the key value is "a*b&c@d", then it gets converted to "a_b_c_d".
      - Upper case characters - They are converted to lower case characters automatically. For example, if the key value is "ABCD", then it gets converted to "abcd".

12. Click **Add tags** to create new user defined tags. The **Add tags** window is displayed. Tags are metadata that is directly persisted back to the actual cloud. They are useful in external system integration through the cloud. Before you use tags, see Prerequisites for tags.

13. Enter the following details of a tag:

    - **Name** - Enter an unique name for the tag. These tags are key value pairs wherein the **Name** is the key.

      Notes:

      - Consider the provider specific limitations when you name the tag or enter a key for it. For more information about IBM Cloud provider naming conventions, see Working with tags.
      - General naming conventions to be followed for the tag name:
        - Only alphanumeric, underscore, and hyphen characters are allowed.
        - Must start with a letter.
    - **End-user permission** - Select **Invisible**, **Read-only**, or **Read-write**.
    - **Type** - Select type from drop-down list. It supports only string type.
    - **Value required** - Determines whether a value is required for the tag.
    - **Specify options** - Select this option for a drop-down list of string values.
    - **Display name** - Enter the display name of the tag.

- **Description** - Enter description of the tag.
- **Default value** - Enter the default value. If you select, **Specify options**, then enter values for the **Label** and **Value** pair fields. Select a record to mark it as default value. You can click the cross mark icon to delete a value record.

14. Click **Add**. The newly created tag gets added to the **Tags** section of the **Overview** tab. You can edit and delete an existing tag from the overflow menu.
15. Click **Add feature** and in the **Features** section to add a service feature. Enter the **Title** and **Description**, then click **Add**. From the overflow menu of an existing feature, you can edit or delete a feature.
16. Open the **Composition** tab and drag the various activities inside the composition canvas.

    To understand the layout and functionality of the **Composition** tab, see Working with Composition tab.
17. Open the **Parameters** tab. From this tab, you can add, edit, and delete an existing parameter. To work with the **Parameters** tab, see Working with the Parameters tab.
18. Open **Plans and form** tab. It has **Plans** and **Plan parameters** sections. Multiple plans are supported. Plans are combination of form or exposed parameters which are overridden to create a form.
19. Click **Add plan**. The **Add plan** window is displayed.
20. Enter the **Plan name** and **Description**.

    For example, create two plans, namely **`VMware`** and **`Powervc`**. There is a **`hypervisordecision`** input parameter in the **Parameters** tab that takes two values:

    ```
    [["value": "VMware"],["value": PowerVC]]
    ```

    Note: The plan name must follow the Kubernetes naming convention, or else the published services do not get listed in the IBM Cloud Pak for AIOps catalog: - Alphanumeric (lower case), underscore, hyphen, and space - Starts with a letter
21. Click **Add**. The newly added plan is added to the **Plans** section. From the overflow menu, you can edit, delete, or preview a plan.

Note: Optionally, you can associate pricing with a plan. However, you have to add pricing in the **Source code**. For more information about the pricing definitions in Service Composition Language, see plans section in Editing a service file page.

19. In the **Plan parameters** section, expand the plan name. All input parameters are listed here so you can override the parameters that are persisted for this plan.

    For example, expand the plan **`VMware`**. The input parameter **`hypervisordecision`** is listed.
20. To override a input parameter, click the overflow menu of a parameter and select **Edit**. The **Edit parameter** window is displayed. Change the permission to invisible and enter a **Default value**.

    For example, enter **Default value** as **`PowerVC`** for the parameter under the **`Powervc`** plan and enter **`VMware`** for the parameter under the **`VMware`** plan. When you create a service instance, the plans are displayed both in IBM Cloud Pak for AIOps and Managed services user interface.
21. Click the **Source code** tab to view, modify, or delete source code for a service.

Note: During service publishing on IBM Cloud Pak for AIOps catalog, if you want to display values for **Details**, **Documentation** and **Support Links** in the **Overview** > **Useful Links** section, then update **`documentationUrl`** and **`supportUrl`** attributes in the **`catalog_metadata`** section of the service template Source Code.

Example:

```
"longDescription": "\nService to deploy EC2 VM (Verified)",
"documentationUrl": "https://www.myibm.com/cam_creating_service.html",
"supportUrl": "http://www.myibm.com",
```

22. Click **Import source file** to import a service file from the local machine. For more information about the service file syntax, see Editing a service file.
23. Save your changes. The **Save** button is disabled if there are validation errors in the service.
24. The service is displayed in the Manage service library. The overflow menu for the service has **Assign access**, **Add Version**, **Duplicate**, **Rename service**, **Change category**, and **Delete**.
25. Expand the service to view the service versions. The overflow menu for a service versions has **Edit**, **Assign custom actions**, **Copy to...**, **Test deploy**, **Publish**, and **Delete**.
26. Test your service by creating a service instance and verifying whether it works correctly. You can deploy services in draft state, but they can be used only for testing purpose. For information about creating a service instance, see Creating a service instance.
27. If the service deployment ends in **`Error`** status, check the deployment logs by clicking the deployed service instance and then the **Log file** tab. When you find out the cause of the error, edit the service by accessing it from the Manage service library, save the changes, and test the service again.

28. If you want to save this service to a Git or Bitbucket repository, see Saving a service to different repositories.

Prerequisites for tags

If you create a service by using a template that includes tags, then the tags are sent to all the deployed instances of the service. Each of the deployed instances in turn passes the tags to the Terraform template. Ensure that your template is capable of consuming tags as Managed services only passes the tags to the Terraform template.

You can use the Managed services to consume the tags.

To use this module, import it and then use it.

For example, if you have packaged the module inline, do the following:

```
module "camtags" {
source = "./Modules/camtags"
}
```

Then the tags can be accessed as a map variable:

```
${module.camtags.tagsmap}
```

or can be accessed as a list:

```
${module.camtags.tagslist}
```

For a sample usage of tags, see the git repository.

For more information about tags module, see Defining the structure of Managed services template.

Working with the Parameters tab

From the **Parameters** tab, you can add new parameters, edit existing parameters, and delete existing parameters.

The tab has **Input Parameters**, **Activity Parameters**, and **Output Parameters** sections. Input and Output Parameters are not defined by any template, but are user-defined parameters to simplify the experience of creating a service instance. They are global variables limited within the scope of this service. You can define them for mapping back values in the activity parameter. Activity Parameters provide the data to deploy a particular template.

- To create new input and output parameters, see Creating new input and output parameters.
- To update or delete an existing input and output parameter, see Managing existing input and output parameters.
- To update an existing activity parameter, see Editing an existing activity parameter

Creating new input and output parameters

From the **Parameters** tab, you can create new parameters for **Input parameters** and **Output parameters**.

Procedure

1. From the drop-down menu, select **Provision action**, **Bind action**, or a custom action.

   Notes:

   - Based on the selection, the parameters change in the **Input parameters** section, **Activity parameters** section, and **Output parameters** section.
   - **Bind action** and custom actions appear in the drop-down list only if created in the **Composition** tab.
2. Click **Create parameter** to add a new input or output parameter. See Creating parameter.

5. Click **Add**. The parameter is added to the appropriate input or output section. You can edit and delete an existing parameter from the overflow menu. You can also verify the parameters in the **Source code** tab.

Creating parameter

1. In the **Create parameter** page, enter the **Parameter key**.

   Note: The naming convention for a parameter key is as follows:

   - Alphanumeric, underscore, and hyphen
   - Starts with a letter
2. Select **End-user permission**.

   The three different types of permissions are as follows:

   - **Invisible** - During parameter creation, enter a value for the **Value**. This parameter is not visible while creating an instance of the service.

     Note: For the Invisible parameter, if you enter incorrect data in the value field, then the Service instance deployment of the content run time fails.

   - **Read-only** - When you create a service instance, you cannot make changes to the value that was defined during creation. Enter or select the following values:

     - **Parameter type** - Select a type. It determines the type of parameter. The available parameter types are string, password, list, boolean, counter, map, connection, and shared parameter.
     - **Display name** - The label or display name of the parameter.
     - **Description** - The description appears as a tooltip when you create the service instance.
     - **Value** - The value of the parameter. When you create the service instance, this parameter is visible to you in read-only format.
     - For the list type, you can click **Add new** to add values to the list.
     - For the map type, you can click **Add new** to add **Key Value** pairs to the map. Click the cross icon to delete a pair.
     - For password types, toggle between **Show** and **Hide** to view or hide the password characters.
   - **Read-write** - When you create a service instance, you can make changes to the value that was defined during creation. Enter or select the following values for **Read-write** permission:

     - **Editable** - If checked, this parameter can be edited while editing a service instance.

     - **Parameter type** - Select a type. It determines the type of parameter. The available parameter types are string, password, list, boolean, counter, map, connection, and shared parameter.

     - **Value required** - Determines whether a value is required for the parameter.

     - **Specify options** - For the string type, you can select this option to create a drop-down list of string values. It is not available for other data types.

     - **Display name** - The label or display name of the parameter.

     - **Description** - The description appears as a tooltip when you create the service instance.

     - **Dynamic** - Enables an input parameter to fetch possible values dynamically in real time depending on the selected values of the dependent parameters.

     - **Pass by name** - Check this if the name should be used instead of the value. If not checked, parameter value is used.

       Note: Parameters of asset type `regions` should not be defined as pass by name. A region should always be passed by a value as per provider implementation.

     - **Source** - A collection of dynamic input parameter implementations. Each source is associated with a cloud provider type.

       For more information about supported providers, see Dynamic parameters supported providers

     - **Asset type** - Select the required asset type from this drop-down list. The asset type list is based on the selection of the **Source**.

     - **Depends on** - These input fields change based on the **Asset type** selection and should be linked to the appropriate input parameters.

       Notes:

       - Two plans with dynamic parameters are not supported yet. They are only supported with the one, standard plan only.
       - Dynamic input parameters must be created sequentially as connection, region, vpc and a subnet.
     - **Validation rule** - The validation rule is an optional field that is available only if you select string or password in **Parameter type** drop-down list. A validation rule (regex) provides an ability to the service composer to restrict the values of an input parameter. If regex is provided during service composition, then the value of the string input parameter must satisfy the regex criteria during service deployment. For example, (\w+?@\w+?\x2E.+). This is not applicable if **Dynamic** is selected.

     - **Validation rule description** - A description that provides information about the validation rule. For example, "Enter a valid email address". Here, (\w+?@\w+?\x2E.+) is the regular expression for an email address. If you enter “abc.xyz.com” as the

value for string input parameters, then service deployment fails due to an invalid email address. This is not applicable if **Dynamic** is selected.

Note: For more information about regex, see regexlib.com.

- **Default value -** Enter a default value. If you select, **Specify options**, then enter values for **Label** and **Value** pair fields. Select a record to mark it as default value. You can click the cross mark icon to delete a value record. This is not applicable if **Dynamic** is selected.

Managing existing input and output parameters

From the **Parameters** tab, you can edit or delete existing input and output parameters.

Procedure

1. Go to the appropriate **Input parameters** or **Output parameters** section.
2. From the overflow menu of the parameter record, click **Edit**.
3. In the **Edit parameters** page, modify the parameter values. For more details on the parameter values, see Creating new input and output parameters.
4. Click **Save**.
5. To delete an existing parameter, select **Delete** from the overflow menu of the parameter record.
6. In the confirmation message, click **Delete**. The parameter record is removed from the list.

Managing an existing activity parameter

In the **Activity Parameters** section, you can update, link, or create a new parameter.

Edit an existing parameter

In the **Edit Parameters** page, you can update values of an existing parameter.

Procedure

1. Go to **Activity parameters** section.
2. Expand the activity and search for your parameter record within it.
3. From the overflow menu of the parameter record, click **Edit**.
4. In the **Edit parameters** page, select the default **Update** tab.
5. Enter or select a new value to update the parameter.
6. Click **Save**.

Link parameter

In the **Edit Parameters** page, link the value to a parameter.

Procedure

1. Go to **Activity parameters** section.
2. Expand the activity and search for your parameter record within it.
3. From the overflow menu of the parameter record, click **Edit**.
4. In the **Edit parameters** page, select the **Link** tab.
5. Select a linking option. You can map parameter values using the **Input parameters** and **Templates** menu options in the **Link parameters** dialog box:

   Note: Not all options are available for a parameter. It varies based on the activities used in the service.

   - **Input parameters** - The **Input parameters** menu option lists all of the parameters that are available in the **Input parameters** section of the main **Parameters** tab of a service.
   - **Shared parameters**

Note: If you link the attribute of a shared parameter, then an input parameter is automatically created. During service deployment, select the dataobject for the datatype. The data attribute value of the selected dataobject is used during provisioning.

- **Templates**
- **Helm Charts**
- **Business Process Manager** - If you have configured a Business Process Manager connection, then all Business Process Manager activities are available for association.

6. Map values with the parameter:
    1. Click to select an option. The values displayed change based on the options that you select and the activities that you have used in your service workflow.

       Notes: For example, if there are two templates A and B placed in sequence in the **Provision action** of the **Composition** tab, then parameters of both the templates are available in the **Activity parameters** section of the **Parameters** tab. If you want to link a parameter of template B, then a list of all parameters of the template A and shared parameters are available for association. If your service involves Business Process Manager activities and helm charts, then they are also available for linking. For the very first activity, template A, you can only link input parameters and shared parameters.
    2. Expand the parameter menu and hover over the parameters to see a plus icon.
    3. Click the plus icon to select it. A green tick mark indicates your selection and also the **Value** is refreshed. You can also use the **Search Parameter** to filter and find a parameter.

Create a new parameter

In the **Edit Parameters** page, you can create service parameter and link to it at the same time.

Procedure

1. Go to **Activity parameters** section.
2. Expand the activity and search for your parameter record within it.
3. From the overflow menu of the parameter record, click **Edit**.
4. In the **Edit parameters** page, select the **Create** tab.
5. Enter a value for the **Parameter key**.
6. Enter or select values for the new parameter.

   Notes:

   - You can only use the **End-user permissions** value of the parameter record from which you opened the **Create** tab. You cannot change the value.

   The three different types of permissions are as follows:

   - **Hidden** - During parameter creation, enter a value for the **Value**. This parameter is not visible while creating an instance of the service.

     Note: For the hidden parameter, if you enter incorrect data in the value field, then the Service instance deployment of the content run time fails.
   - **Read-only** - When you create a service instance, you cannot make changes to the value that was defined during creation. Enter or select the following values:
     - **Parameter type** - This value cannot be changed. It is set to the parameter type of the parameter you are editing.
     - **Display name** - The label or display name of the parameter.
     - **Description** - The description appears as a tooltip when you create the service instance.
     - **Value** - The default value of the parameter. When you create the service instance, this parameter is visible to you in read-only format.
     - For the list type, you can click **Add new** to add values to the list.
     - For the map type, you can click **Add new** to add **Key Value** pairs to the map. Click the cross icon to delete a pair.
     - For the password type, toggle between **Show** and **Hide** to view or hide the password characters.
   - **Read and Write** - When you create a service instance, you can make changes to the value that was defined during creation. Enter or select the following values:
     - **Parameter type** - This value cannot be changed. It is set to the parameter type of the parameter you are editing.
     - **Value required** - Determines whether a value is required for the parameter.

- **Specify options** - For the string type, you can select this option to create a drop-down list of string values. It is not available for other data types.
- **Display name** - The label or display name of the parameter.
- **Description** - The description appears as a tooltip when you create the service instance.
- **Default value** - The default value.

7. Click **Save** to create and save the new parameter.

Working with the Composition tab

The Service Composition tab has the following sections:

- A palette to drag the different service components.
- A canvas where you can draw your workflow.
- A Parameter details pane that changes based on the activity you select in the canvas.

Palette of service components

The palette includes templates, helm charts, flow components, notifications, and integration components that you can drag and drop on the canvas.

To work with the different service components, see Service components in the Composition tab. For details about the supported interpolation expression of these service components, see Supported interpolation expressions.

Canvas in the middle

The canvas has the following sections within it:

- **Provision action** and **Create action** actions

  When you create a service, you can define actions for the service from the composition tab. The actions drop-down list of the canvas includes **Provision action** and **Create action** options. You can create a **bind** action or **custom** actions by using the **Create action** option.

  Note: The **Provision action** is displayed by default in the action list.

  The **Provision action** provisions a resource instance. Optionally, if you want to execute some additional bind actions on a provisioned resource instance, you can create a **bind** or **custom** action.

  Note: The **bind** and custom actions you define are applicable across all plans defined for that service.

  Steps to create a action:

  1. In the action drop-down, select **Create action** to create a new action. By default, **Provision action** is selected in the **Composition** tab.
  2. In the **Type** drop-down, select the type of action.
     1. **bind**: The **Name** of the action is "Bind" and can not be modified.
     2. **custom**: Enter the name of the action in **Name** text box. The name can not be **bind** for the custom action.
  3. Enter the **Description** for the action.
  4. Click **Create**. You can create only one bind action. If you try to create a second bind action, the **Create** button is disabled. There is no limit for creating custom actions.
  5. You can now include a Terraform templates or other activities in the bind canvas. These activities run on a deployed instance of a service. It is recommended that you include only Terraform templates that do not deploy a resource instance. You can view the details of the bind and provision in the **Source code** tab.

     Note: You can also create and invoke stand-alone action independent of any service or service instance. For more information on the APIs, see Create Action and Invoke Action in the Common Managed services APIs page.

- **Primary flow** and **Error flow**

  The **Primary flow** and **Error flow** sections are positioned below the action drop-down list in the canvas. You can have primary flow and error flow for **Provision**, **bind**, and **custom** actions.

  The **Primary flow** is initiated when you initially request a service. You can drag the service components of the workflow here. In the canvas they are generally termed as "activities."

The **Error flow** is initiated when an error occurs in the primary flow during deployment. Similar to the primary flow, you can drag service components from the palette.

Notes:

- You can collapse and expand the **Primary flow** and **Error flow** sections in the canvas.
- You can use retry or resume for failed deployments. For the procedure to retry or resume, see Resuming or retrying a service instance.
- The instance details and the activity log section of the **Log file** tab do not show details of the activities executed in the error flow. In the **Log file** tab, the error flow execution details are reported in the **ON ERROR FLOW** part of the service log section.
- The error flow is intended to notify you through email or slack or update a service now ticket about the error. As a result, it is not recommended to provision new resources in this flow. If new resources are provisioned, then the instances from the error flow do not get terminated during the 'Terminate' action. Also, an error that occurs in the error flow sequence is not handled separately.
- In error flows, for more information about interpolations that can be used to identify details around the activity that failed, see Service instance level parameters.
- You can use the OR (||) operator for the service activities in the error flow.
- If you use interpolation in the error flow around a non executed decision from the primary flow, the resultant value is an empty string.
- Limitations in the **Error flow** section:
  - You cannot use a decision service component in the **Error flow** section.
  - Whenever a service deployment fails in Terraform plan, the error flow is not executed. It is executed only when the Terraform plan gets executed successfully and some failure occurs during apply.

Parameters section

The parameters on the **Composition** tab vary based on the current activity that is selected in the canvas. To work with the activity parameters, see the following topics:

- Service components in the Composition tab
- Supported interpolation expressions

Additional features

- You can hover over an activity in the canvas to clone or delete. Additionally, for a decision activity, you can collapse the activity view in the flow.

  Note: Before you delete a component activity, ensure that it is not linked to other activities in the flow. If it is linked, then manually update all mapping either in the **Composition** tab or the **Source** tab, otherwise, the service breaks.

- You can use the control buttons to search the canvas, fit-to-screen, and zoom.

- If there are no errors or warnings in your service, then the page displays a **No issues** message. Whenever errors and warnings occur, a notification panel is displayed with the count of errors and warnings. Click for more information about the context of errors and warnings. In addition, in the composer canvas, a small icon (error and/or warning) is displayed against the related activity that has a validation problem. When you click **Save**, the API validation messages are displayed, however, they go away whenever you try to modify the service or move to other tabs. For more information about service validations, Managed services validation approaches, and behavior, see Service validations.

Activities in the Composition tab

Flow components, notifications, integration components, applications, templates and Helm charts are the different service components that you can work with in the **Composition** tab.

- Flow components
  - Decision
- Email notification
  - Email notification
  - Configuring and managing SMTP server connection for Email notifications
- Integration components
  - Rest Hook
- Applications
- Templates
- Helm charts
- Ansible
  - Credential
  - Inventory
  - Job Templates

For input and output parameter mapping for these activities, see Supported interpolation expressions.

For troubleshooting and limitations in the service composer, see Troubleshooting Service Composer or Portal issues.

Decision

In the **Composition** tab, the decision component is available underneath the **Flow components** menu. It is used for the purpose of selecting a flow based on a decision.

Procedure

1. Drag **Flow Components** > **Decision** in the composition canvas.
2. Select or enter the following details:
    - **Title** - Enter the title of the decision activity. The **Title** must have a unique value.
    - **Description** - The description of the decision activity.
    - **Mapped Parameters** - Click **Link parameter** to map an existing parameter or create a new parameter. Alternatively, click **Add parameter** and enter an interpolation value. Click the blue tick mark to save the value.
        - Steps to map an existing parameter or create a new parameter of type mapped parameter:
            - Click **Link parameter** in the **Activity details** pane to map a new parameter.
            - In the **Link parameter** dialog, click **Link** to map an existing parameter.
            - Map an existing parameter or create a new parameter and link it automatically:
                - Link parameters:
                    - Select a linking option. You can map parameter values using the **Input Parameters** and **Templates** menu options in the **Link Parameter** dialog box:
                        - **Input Parameters** - This menu option lists all string parameters that have **Options required** set to true in the **Input Parameters** section of the main **Parameters** tab of a service.
                        - **Templates** - This option lists all parameters of all template activities that are available prior in sequence to this decision activity.

                      The values displayed in the **Activity details** pane change based on what you select and which activities you use in your service workflow.

                      Note: For example, if there are two templates A and B placed in sequence in the **Provision Action** of the **Composition** tab, then parameters of both the templates are available in the **Activity Parameters** section of the **Parameters** tab. If you want to link a parameter of template B, then a list of all parameters of the template A and shared parameters are available for association. If your service involves Business Process Manager activities and helm charts, then they are also available for linking. For the very first activity, template A, you can only link input parameters and shared parameters.
                    - Expand the parameter menu and hover over the parameters to see a plus icon.
                    - Click the plus icon to select it. A green tick mark indicates your selection and also the **Value** is refreshed. You can also use the **Search Parameter** to filter and find a parameter.
                    - Click **Save**.
                    - You can also click **Remove Link** in the **Link Parameter** dialog box to remove a mapping.
                - Create a new parameter and link it at the same time:
                    - In the **Link Parameter** dialog box, click **Create**.
                    - Enter a value for the **Parameter Key**.
                    - Enter or select values for the new parameter.

                      Notes:
                        - You can only use the **End-User Permissions** value of the parameter record from which you opened the **Create New** tab. You cannot change the value.
                        - You cannot change the **Parameter Type** value. It is dependent on the parameter record type from which you opened the **Create New** tab.
                    - Enter or select the following values for **Read and Write** permission:

- **Parameter Type** - This value cannot be changed and is set to the parameter type of the parameter from which you opened the **Link parameter** dialog.
- **Value Required** - Determines whether a value is required for the parameter.
- **Specify Options** - For string data type, you can select this option for a drop-down list of string values.
- **Display Name** - The label or display name of the parameter.
- **Description** - The description appears as a tooltip when the customer orders this service.
- **Default Value** - Enter values for **Label** and **Value** pair fields. Select a record to mark it as default value. You can click the cross mark icon to delete a value record.

- Click **Save** to create and save the new parameter.

- **Cases** - The value of **Cases** depends on the selected **Mapped parameter**.

  An example of **Cases** can be `case 0: VMware` and `case 1: IBM Cloud`.

Email notification

In the **Composition** tab, the **Email Notification** is available underneath the **Notification** menu.

It triggers an email notification at the specified point in the flow. Before using email notification, you must configure a connection to the SMTP server. For more information, see Configuring SMTP server connection for Email notifications using Managed services console and Configuring a connection for notifications using API.

Procedure

1. Drag **Notification** > **Email Notification** in the composition canvas.
2. Check **Editable** if you want to edit these parameters while editing the service instance.
3. Enter values for **Title**, **To**, **Cc**, and **Content**.

   Note: If running behind a proxy server, make sure the proxy SMTP server is configured and its values are set in the email configuration.
4. Optionally, in **Body**, enter the input value for the Email Notification activity in JSON format.

   Tip: The assistive user interface provides suggestions when you type "$". You can select from the interpolation options, such as ${input_parameters, ${svc_instance, ${datatypes. After you select the interpolation option, enter "." for the user interface to list all probable input parameters that are defined prior to this email notification activity. This feature is available for **To**, **Cc**, **Content**, and **Body**.

Managing Email Configuration

You can configure primary and secondary SMTP Email servers and manage their configurations.

Managing Email Configuration:

- Configure Email Server
- Searching SMTP Configuration
- Editing SMTP Configuration
- Deleting SMTP Configuration

Note: If you close the window without saving during create or manage operations, then the values are reset to empty.

Configure Email Server

From the **Email configuration** page, you can create an email server configuration.

1. In the navigation bar, click **Automate Infrastructure** > **Manage**. Click the **Email configuration** tab.
2. Click **Create configuration**.
3. Select either of the following options in the **Assign access** section:

   - **Make email server globally accessible (available to all users)** - Email servers created globally are accessible by all users.
   - **Make connection part of a namespace** - Associate the email server with a specific namespace. For this option, select a namespace from the drop-down list to which the email server will be assigned.

Note:

- Based on the value selected in the namespaces drop-down list on the **Email configuration** main page, the **Assign access** and namespace values are automatically selected. Modify the values if you want.
- If there is already a configuration available for the selected namespace, then those details are automatically pre-filled.

4. Select **Primary** or **Secondary** type.
5. Enter the following details of the configuration:
   - **SMTP hostname** - Enter the host name of the SMTP server.

     Note: If running behind a proxy server, enter the proxy server host name or IP address.
   - **SMTP port** - Enter the port of the SMTP server.
   - **SMTP username** - Optionally, enter the user name of the SMTP server. **Note:** Whenever an email notification is sent, the **SMTP username** is used as the sender email address.
   - **SMTP password** - Optionally, enter the password of the SMTP user.
   - **Test email recipient** - Optionally, enter the email address to which a test email will be sent, if requested. **Note**: To send a test email, click the **here** link. A message appears confirming the success or failure of the test email.
6. Click **Save**.
7. SMTP server details are shown in the **Email server configurations** table.

Searching SMTP Configuration

From the **Email configuration** page, you can search for an email server configuration.

1. In the navigation bar, click **Automate Infrastructure** > **Manage**. Click the **Email configuration** tab.
2. Search for your email server configuration by using any of the following search filters:
   - Enter the host name in **Search email configurations**.
   - Click the **Namespace** column heading to sort the table by namespace.
   - At the bottom of the table, you can use the **Email configurations per page** drop-down to select the number of records you want to display in a page at a time. Use the **<** and **>** to navigate to the previous and next page.
   - You can select a namespace from the namespaces drop-down list. The list of email server configurations change based on the selected namespace. Along with the records for the selected namespace, all globally accessible records are also shown in the list. You can also select **All namespaces** to see all the configurations for which you have access. Each namespace can have a maximum of two configurations.

Editing SMTP Configuration

From the **Email configuration** page, you can edit an email server configuration.

1. In the navigation bar, click **Automate Infrastructure** > **Manage**. Click the **Email configuration** tab.
2. In the **Email server configurations** table, click the overflow menu of your SMTP server.
3. Click **Edit**. **Note**: You can also click the servers hostname to edit it.
4. The **Assign access** and **Type** are read-only and cannot be changed.
5. If you want to send a test email, provide a **Test email recipient** and click the **here** link. A test email is sent to this email address. A message appears confirming the success or failure of the test email.
6. Modify the details and **Save**.

Deleting SMTP Configuration

From the **Email configuration** page, you can delete an email server configuration.

1. In the navigation bar, click **Automate Infrastructure** > **Manage**. Click the **Email configuration** tab.
2. In the **Email server configurations** table, click the overflow menu of your SMTP server.
3. Click **Delete**.
4. In the **Remove email server** message box, click **Delete** to confirm the deletion.

Rest Hooks

In the **Composition** tab, the **Rest Hook** option is available within the **Integration** menu. It is used to make a REST call to an external system.

Procedure

Note: If running behind a proxy server, make sure the proxy environment variables (http_proxy, https_proxy) are set during the installation of Managed services.

1. Drag **Integration** > **Rest Hook** into your flow.
2. Check **Editable** if you want to edit these parameters while editing the service instance.
3. Enter the following values:
   - **Title** - Enter the name or the title for the Rest Hook.
   - **Activity ID** - The read-only activity ID is automatically generated.
   - **Input URL** - Enter the URL where the payload is delivered.

     Tip: The assistive user interface provides suggestions when you type "$". You can select from the interpolation options, such as ${input_parameters, ${datatypes. After you select the interpolation option, enter "." for the user interface to list all input parameters defined before this Rest Hook activity was created.
   - **Method** - Select the **Method**. The supported values are GET, POST, PUT, PATCH, and DELETE.
   - **Payload** - Enter the request body in JSON format. Quotes must be added for parameters with type String or Password. For Counter, List, and Map parameters, you must stringify the JSON data if a parameter value is a linked parameter.

     For example, name is a string parameter and amount is not:

     ```
     {
       \"name\": \"${param.name}\",
       \"amount\": "${param.amount}"
     }
     ```

     Tip: The assistive user interface is available for **Payload**. It enables you to easily enter multiple interpolations. Type "$" for the assistive user interface to provide you suggestions when you want to add an interpolation.
   - In the **Header/Auth** section, enter the following values:
     - The **Key** and **Value** are displayed for existing header keys and values. Click **Edit** to add or delete header keys. You cannot edit an existing header.
     - **Auth Type** - Select the authentication method. `basic` and `token` authentication methods are supported.
     - **Auth Username** - If you select `basic` authentication, enter the user name to be used for authentication with the Rest Hook authentication management system.

       Tip: The assistive user interface enables you to associate an input parameter with the user name so you can retrieve the user name dynamically during runtime. Type "$" for the assistive user interface to provide you suggestions.
     - **Auth Password** - If you select `basic` authentication, enter the password for the user.
     - **Bearer token** - If you select `token` authentication, enter a bearer token for the Rest Hook authentication management system.
4. Open the **Destroy** tab and enter values to define the Rest Hook that is run when the service instance is terminated. The parameters in the **Destroy** tab are the same as those in the **Create** tab.

For supported interpolations, see Input and output parameter mapping for REST Hook.

Applications

In the **Composition** tab, the published applications are displayed under the **Applications** menu.

Procedure

1. Browse through the list of published application or use the **Filter** to search for a particular application.
2. Drag the selected application to the pane.

3. Check **Editable** flag if you want to edit these parameters while editing the service instance.
4. Select or enter the following values for **Basic information**:
    - **Title** - Enter title of the application activity. It is a mandatory field. It is also the label name that appears in the application activity.
    - **Activity ID** - The ID is automatically generated and is not editable. Click **Copy** to copy the value of the activity ID to your clipboard.
    - **Template name** - It is the name of the application and is not editable.
    - **Template provider** - It is IBM Cloud Pak for AIOps and is not editable.
    - **Version** - It is `1.0` and is not editable.
    - **Instance name** - Enter the instance name of the application activity.
5. Update the details in **Parameters** tab. For more information, see Parameters tab of an activity.

Templates

In the **Composition** tab, **Templates** has a list of Terraform templates categorized based on the template provider. The number in parenthesis next to the provider name indicates the number of templates available for that provider.

Procedure

1. Browse through the list of Terraform templates or use the **Filter** to search for a particular template.
2. Drag a template into your flow. After the template has loaded you can set its parameter values.

   Note:
    - If you want to use IBM Business Process Manager (BPM) templates, you must have activated them during installation. For information about activating these templates after installation see Enabling or disabling IBM Business Process Manager.
    - This list contains templates that are globally accessible and templates that belong to the services namespace.
3. Check **Editable** if you want to edit these parameters while editing the service instance.
4. Select or enter the following values for **Basic information**:
    - **Title** - Enter title for the template activity.
    - **Activity ID** - The ID is automatically generated and is read-only.
    - **Template name** - The name of the template. This field is read-only.
    - **Template provider** - The name of the provider. This field is read-only.
    - **Version** - Select the version of the template. The templates default version is preselected.

      Note: If you switch versions, the service parameters are overwritten with the values from the selected version.
    - **Terraform version** - Select the Terraform version used to deploy the template.
    - **Use compatible version** - By default, this is checked. During deployment, if the selected Terraform version is not found, then the first compatible Terraform version is used based on the constraints that are defined in the template. You can uncheck this option if you want to use a specific Terraform version.
    - **Instance name** - Enter the instance name of the template activity.
    - **Connection** - Connections that are globally accessible or in the same namespace as the service are available for selection. There are different ways to enter values for the connection:
        - Select an existing connection from the drop-down list. Support is extended for IBM Cloud Pak for AIOps connections and connections to an external IBM Cloud Kubernetes Service.
        - You can create dynamic connections where the connection details are fetched at runtime. The output of a previous activity in the flow can be mapped to the connection field as follows:

          `${activity.id.output.paramname}`

            - To link the connection to an existing parameter:
                - Click **Link parameter** in the **Activity details** pane to map a new parameter.
                - In the **Link parameter** dialog, click **Link**.

- Click **Input Parameters** to see a list of parameters available in the **Input parameters** section of the main **Parameters** tab.
- Click **Templates** to see a list of parameters from all template activities in the flow before this template.

  The values displayed change based on what options you select and which activities you have used in your service workflow.

  Note: For example, if there are two templates A and B placed in sequence in the **Provision action** of the **Composition** tab, then parameters of both templates are shown in the **Activity parameters** section of the **Parameters** tab. If you want to link a parameter of template B, then the parameters of template A and shared parameters are available for linking. If your service includes Business Process Manager activities or Helm charts, then they are also available for linking. For the first activity, template A, you can only link to service level parameters and shared parameters.
- You can use **Search parameter** to filter and find parameters.
- Hover over a parameter to see a plus icon.
- Click the plus icon to select it. A green tick mark indicates your selection and the **Value** is refreshed.
- Click **Save**.

- To create a parameter and link it at the same time:
  - In the **Link parameter** dialog, click **Create**.
  - Enter a value for the **Parameter key**.
  - The **End-user permission** is set based on the permission of the parameter being linked.
  - Enter the parameter values:
    - **Editable** - Check the box if you want to edit these parameters while editing the service instance.
    - **Parameter type** - Set to `connection` and is read-only.
    - **Display name** - The label or display name for the parameter.
    - **Default value** - Optionally, select a cloud connection. If you do not select a value during creation time, you must specify it during deployment.
    - **Cloud provider** - Set to the cloud provider for the template and is read-only.
    - **Description** - The description appears as a tooltip for this parameter when the customer orders this service.
  - Click **Save** to save the new parameter and link it to your connection.

  Note: For an IBM Business Process Manager (BPM) connection, you do not have create a connection explicitly in the **Manage** > **Cloud connections** page. You can directly select the "BPM Connection" value from the **Connection** drop-down list. You do not have to provide any credentials for the connection as the service deployment gets the IBM BPM connection details from environment variables.

- **Content runtime** - For middleware templates, you must enter values for content runtime. It is a required value.

  Note: For middleware templates, you can select an Automated Content Runtime that is not deployed on the same cloud provided proper network connectivity exists.

  Click **Link parameter** in the **Activity details** pane to link the value to a parameter:

  - To link to an existing parameter:
    - In the **Link parameter** dialog, click **Link**.
    - Select a linking option. You can link parameter values using the **Input parameters** and **Templates** menu options in the **Link Parameter** dialog:
      - **Input parameters** - The list of parameters available in the **Input parameters** section of the main **Parameters** tab
      - **Templates** - The list of parameters from all template activities in the flow before this template.

        The values displayed change based on what options you select and which activities you have used in your service workflow.

        Note: For example, if there are two templates A and B placed in sequence in the **Provision action** of the **Composition** tab, then parameters of both the templates are available in the **Activity parameters** section of the **Parameters** tab. If you want to link a parameter of template B, then a list of all parameters of the template A and shared parameters are available for association. If your service involves Business Process Manager activities and Helm charts, then they are also available for linking. For the first activity, template A, you can only link input parameters and shared parameters.
    - Expand the parameter menu and hover over a parameter to see a plus icon.

- Click the plus icon to select it. A green tick mark indicates your selection and the **Value** is refreshed.
- Click **Save**.
- You can also click **Remove link** in the **Link parameter** dialog box to remove a mapping.

- To create a parameter and link it at the same time:
  - In the **Link parameter** dialog, click **Create**.
  - Enter a value for the **Parameter key**.
  - The **End-user permission** is set based on the permission of the parameter being linked.
  - Enter the parameter values:
    - **Parameter type** - Set to `sharedparameter` and is read-only.
    - **Display name** - The label or display name for the parameter.
    - **Data type** - Set to `Advanced content runtime` and is read-only.
    - **Description** - The description appears as a tooltip for this parameter when the customer orders this service.
    - **Value** - Select a Data Object.
  - Click **Save** to create and save the new parameter.

- If any of the Terraform templates have shared parameters in their definition, then the Data Types are displayed in the **Basic information** tab. You must select values for the Data Objects that are associated with the Data Types.

  Instead of hard coding the value of a Data Object, link shared parameter values to parameters. To create a parameter:

  - Create a parameter in the **Parameters** tab with type `String`.
  - In the **Source code** tab, change the type to `sharedparameter` and provide a value for `datatype`. For example, this parameter has type `sharedparameter` and is associated to the `bastionhost` Data Type.

    ```
    "input_parameters": [
     {
      "name": "shared",
      "label": "shared",
      "customtype": "string",
      "type": "sharedparameter",
      "datatype": "bastionhost",
      "immutable": false,
      "hidden": false,
      "required": false,
      "secured": false
      }
     ],
    ```

  - For the Data Object value in the **Parameters** section of the **Composition** tab, you can enter a dynamic expression:

    The format of the expression is as follows:

    ```
    ${input_parameters.shared}
    ```

    For the behavior of this dynamic Data Object, see Creating a service instance. You must define a default Data Object whenever the Data Type has multiple Data Objects.

    Click **Link parameter** in the **Activity details** pane to link the value to a parameter:

    - To link to an existing parameter:
      - In the **Link parameter** dialog, click **Link**.
      - Select a linking option. You can link parameter values using the **Input parameters** and **Templates** menu options in the **Link Parameter** dialog:
        - **Input parameters** - The list of parameters available in the **Input parameters** section of the main **Parameters** tab
        - **Templates** - The list of parameters from all template activities in the flow before this template.

        The values displayed change based on what options you select and which activities you have used in your service workflow.

        Note: For example, if there are two templates A and B placed in sequence in the **Provision action** of the **Composition** tab, then parameters of both the templates are available in the **Activity parameters** section of the **Parameters** tab. If you want to link a parameter of template B, then a list of all parameters of the template A and shared parameters are available for association. If your service involves Business Process Manager activities and helm charts, then they are also available for linking. For the very first activity, template A, you can only link input parameters and shared parameters.
      - Expand the parameter menu and hover over a parameter to see a plus icon.

- Click the plus icon to select it. A green tick mark indicates your selection and the **Value** is refreshed.
- Click **Save**.
- You can also click **Remove Link** in the **Link parameter** dialog to remove a mapping.

- To create a parameter and link it at the same time:
  - In the **Link parameter** dialog, click **Create**.
  - Enter a value for the **Parameter key**.
  - The End-user permission is set based on the permission of the parameter being linked.
  - Enter the parameter values:
    - **Parameter type** - Set to `sharedparameter` and is read-only.
    - **Display name** - The label or display name of the parameter.
    - **Data type** - The Data Type value is preselected. You cannot change the value.
    - **Description** - The description appears as a tooltip for this parameter when the customer orders this service.
    - **Value** - Select a Data Object.
  - Click **Save** to create and save the new parameter.

For information about shared parameters in interpolation expressions, see Using shared parameters in service interpolations and Data Types in Service Composition Language.

For information about Data Types and Data Objects, see Shared parameters.

For managing shared parameters from Managed services, see Managing Data Objects from the user interface.

5. Update the details in **Parameters** tab. For more information, see Parameters tab of an activity.

Notes:

- If you use a long running BPM template in your service that requires claim and approvals by the administrator of Business Process Manager, include the following parameters in the source code tab as follows:

```
"async_req": true
"long_running_delay": "<time in minutes>",
"long_running_timeout": "<time in hour>"
```

The async_req field is not shown in the **Parameters** tab. To change its value use the **Source code** tab.

For more details, see Example of a Business Process Manager template in Service Composition Language Structure topic.

- If a variable in a Business Process Manager template is not Boolean or String, then Managed services considers it as a String. For complex type input parameters, provide the JSON as string. For example:
  - Template input parameters in Business Process Manager:

```
{
  "inputString": {
    "type": "String",
    "isList": false
  },
  "inputInteger": {
    "type": "Integer",
    "isList": false
  },
  "inputBoolean": {
    "type": "Boolean",
    "isList": false
  },
  "parent": {
    "type": "ParentData",
    "isList": false
  }
}
```

  - Template input parameter values in Managed services:

```
inputString = Bob
inputInteger = 555
inputBoolean = false
parent = {"Name": "David", "Child": {"id": 21, "Name":"John"}}
```

Helm charts

The **Helm** section has a list of Helm charts available from the underlying IBM Cloud Pak for AIOps. If you want charts in the remote IBM Cloud Pak for AIOps to be available in Managed services, copy your charts to the local IBM Cloud Pak for AIOps.

The number mentioned within parenthesis next to the type indicates the number of Helm charts available from the underlying IBM Cloud Pak for AIOps. The number of charts visible is governed by the access defined by the IBM Cloud Pak for AIOps team for your user ID.

For a list of limitations, see the Helm limitations section of Troubleshooting Service Composer or Portal issues.

Procedure

1. Browse through the list of Helm charts or use the **Filter** to search for a particular Helm chart.
2. Drag a Helm chart into your flow. After the Helm chart has loaded you can set its parameter values.

   Note: The listed Helm charts do not have the IBM Cloud Pak for AIOps icons.
3. Check **Editable** if you want to edit these parameters while editing the service instance.
4. In the **Basic Information** tab, enter or select the following details:

   - **Title** - Enter a title for the Helm chart activity. It is a required field. It is also the label name that appears in the Helm chart activity.
   - **Activity Id** - The read-only ID is automatically generated.
   - **Helm Chart Name** - The name of the Helm chart. This field is read-only.
   - **Helm Chart Provider** - The name of the provider. This field is read-only.
   - **Version** - Select the Helm chart version. The Helm charts default version is preselected.
   - **Release Name** - Enter the release name of the Helm chart activity.
   - **Connection** - There are different ways to enter values for the connection:
     - If you want to select an existing connection, select a connection from the drop-down list. Support is extended for local IBM Cloud Pak for AIOps (MCM Connection) connection and connections to external IBM Cloud Kubernetes Service.
     - You can create dynamic connections where the connection details are fetched during runtime. The output of a previous activity in the sequence can be mapped to the connection field as follows:

       `${<activity>.<id>.output.<param-name>}`

       or to a service parameter: `${input_parameters.paramname}`
   - **Chart License** - Select **I have read and agreed to the license agreements**.
5. Update the values in the **Parameters** tab. All parameters defined in the Helm chart are displayed. For more information, see Parameters tab of an activity.

Ansible

In the **Composition** tab, the Ansible component is available underneath the **Integration** menu.

- Credential
- Inventory
- Job Templates

Ansible Credential

Ansible credentials are utilized by Tower for authentication when launching jobs against machines.

After the service with credential activity is deployed on the Managed services, the credential is created in the Ansible.

Procedure

1. Select or enter the following values for **Basic information**:

   - **Title** - Enter title of the credential activity. It is a mandatory field. It is also the label name that appears in the credential activity.
   - **Activity Id** - The Id is automatically generated and is not editable. Click **Copy** to copy the value of the activity Id to your clipboard.
   - **Template name** - It is the name of the template and is not editable.
   - **Template provider** - It is the name of the provider and is not editable.

- **Version** - Select the template version. A template is always at its default version.

  Note: If you select another version and again switch back to the original version, then service parameters gets overwritten by the values from the original template version.

- **Instance name** - Enter the instance name of the template activity.
- **Connection** - There are different ways to enter values for the connection:
  - Steps to map an existing parameter or create a new parameter of type connection:
    - Click **Link parameter** in the **Activity details** pane to map a new parameter.
    - In the **Link parameter** dialog box, click **Link** to map an existing parameter.
    - Map an existing parameter or create a new parameter and link it automatically:
      - Link parameters:
        - Select a linking option. You can map parameter values using the **Input parameters** and **Templates** menu options in the **Link Parameter** dialog box:
          - **Input parameters** - The **Input parameters** menu option lists all of the parameters that are available in the **Input parameters** section of the main **Parameters** tab of a service.
          - **Templates** - This option lists all parameters of all template activities that are available prior in sequence to this template activity.

          The values displayed in the in the **Activity details** pane change based on what you select and which activities you use in your service workflow.

          Note: For example, if there are two templates A and B placed in sequence in the **Provision Action** of the **Composition** tab, then parameters of both the templates are available in the **Activity parameters** section of the **Parameters** tab. If you want to link a parameter of template B, then a list of all parameters of the template A and shared parameters are available for association. If your service involves Business Process Manager activities and helm charts, then they are also available for linking. For the very first activity, template A, you can only link input parameters and shared parameters.
        - Expand the parameter menu and hover over the parameters to see a plus icon.
        - Click the plus icon to select it. A green tick mark indicates your selection and also the **Value** is refreshed. You can also use the **Search parameter** to filter and find a parameter.
        - Click **Save**.
        - You can also click **Remove Link** in the **Link parameter** dialog box to remove a mapping.
      - Create a new parameter and link it at the same time:
        - In the **Link parameter** dialog box, click **Create**.
        - Enter a value for the **Parameter Key**.
        - Enter or select values for the new parameter.
        - When you create a service instance, you can make changes to the values that were defined during creation. Enter or select the following values for **Read and Write** permission:
          - **Parameter type** - You cannot change the parameter type. It is connection by default.
          - **Display name** - The label or display name of the parameter.
          - **Description** - The description appears as a tooltip when the customer orders this service.
          - **Cloud provider** - The cloud provider is pre-selected. You cannot change the value.
          - **Default value** - Optionally, enter the value for connection. If you do not provide the value during design time, you must add it during deployment.
        - Click **Save** to create and save the new parameter.
  - You can create a parameter of type string in the **Parameter** tab. In the **Source code** tab, change its type to connection and add provider value as shown in the following example:

```
"input_parameters": [
 {
 "name": "Conn",
 "label": "Conn",
 "Customtype": "string",
 "type": "connection",
 "provider": "VMware vSphere",
 "immutable": false,
 "hidden": false,
 "required": false,
 "secured": false
  }
 ]
```

In **Connections**, enter `${input_parameters.Conn}`. For the behavior of this dynamic cloud connection at runtime, see Creating a service instance.

- The connection parameter type is not applicable for "Other" type of cloud connections because it is globally available.

2. Update the details in **Parameters** tab. For more information, see Parameters tab of Ansible Credentials activity.

   If a parameter has Read-Write user permission, then you can edit the values to override existing permission and parameter values.

Note: If the **Editable** flag is set to FALSE, this service does not contribute in the service instance deployment.

Parameters tab of Ansible Credential activity

You can set the parameter values for Ansible credentials from the **Composition** tab > Credential activity > **Parameters** tab.

Procedure

1. Select the activity in the **Composition** tab. The input parameters are displayed along with their name and value in your activity in the **Activity details** pane.

   Note: The Managed services and Ansible integration supports only the *Machine* credential type.

2. Enter the following values:

   - **credential_name_suffix**: Suffix is used while creating the credential on the Tower. Credential is created as ActivityInstanceName_suffix.
   - **user_name**: The user name for the credential.
   - **ssh_private_key**: The key for the credential.
   - **password**: The password for the credential.

   Note: Enter either the **ssh_private_key** or **password** or both for the credential.

3. Use **Search parameters** to filter and search for your parameter.

4. You can hard code the value of a parameter, link the value to a parameter or create a new parameter:

   - Link the parameter value:
     - Select a parameter. The following menu options are available for mapping the parameter values:

       Note: Not all options are available for a parameter. It varies based on the activities used in the service.

       - **Input parameters**: The **Input parameters** menu lists all of the parameters that are available in the **Input parameters** section of the main **Parameters** tab of a service.
       - **Shared parameters**

         Note: If you link the attribute of a shared parameter, then an input parameter is automatically created. During service deployment, select the dataobject for the datatype. The data attribute value of the selected dataobject is used during provisioning.

     - Map the parameter values:
       - Expand the parameter menu and hover over the parameters to see a plus icon.
       - Click the plus icon to select it. A green tick mark indicates your selection and also the **Value** is refreshed. You can also use the **Search parameter** to filter and find a parameter.
   - Create a new parameter:
     - Click **Link parameter** in the **Activity details** pane to map a new parameter.
     - In the **Link parameter** dialog box, click **Create**. You can create only a new parameter whose type exists in the template activity. For example, a template has string, Boolean, and map parameter types. If you want to create a new parameter of type counter, you must go to the main **Parameters** tab. To work with the main **Parameters** tab, see Working with the Parameters tab.
     - Select **End-User Permissions**.

       The three different types of permissions are as follows:

- **Invisible** - During parameter creation, enter the following value:
  - **Default value**. This parameter is not visible while creating an instance of the service.

Note: For the Invisible parameter, if you enter incorrect data in the default value field, then the Service instance deployment of the content run time fails.

- **Read-only** - During parameter creation, enter or select the following values:

  - **Parameter type** - Select the type of parameter. The available parameter types are string, password, list, Boolean, counter, map, connection, and shared parameters.
  - **Display name** - The label or display name of the parameter.
  - **Description** - The description appears as a tooltip when the customer orders this service.
  - **Default value** - The default value of the parameter. When you create the service instance, this parameter is visible to you in read-only format.

  For list data type, you can click **Add New** to add a list of values.

  For map data type, you can click **Add New** to add a list of **Key Value** pairs. Click cross icon to delete a value record.

  For password data type, toggle between **Show** and **Hide** to view or hide the password characters.

  - **Read and Write** - When you create a service instance, you can make changes to the values that were defined during creation. Enter or select the following values for **Read and Write** permission:
- **Parameter type** - Select a type. It determines the type of parameter. The available parameters are string, stringlist, password, list, boolean, counter, and map.
- **Value required** - Determines whether a value is required for the parameter.
- **Specify options** - For string data type, you can select this option for a drop-down list of string values.
- **Display name** - The label or display name of the parameter.
- **Description** - The description appears as a tooltip when the customer orders this service.
- **Validation rule** - The validation rule is a non-mandatory field that is available only if you select "string" or "password" value in **Parameter type** drop-down list. Validation Rule (regex) provides an ability to the service composer to restrict the values of an input parameter. If regex is provided during service composition, then the value of the string input parameter must satisfy the regex criteria during service deployment. For example, (\w+?@\w+?\x2E.+).
- **Validation rule description** - Validation Rule Description (regexdesc) provides information about the validation rule. For example, “Enter a valid email address”. Here, (\w+?@\w+?\x2E.+) is the regular expression for an email address. If you try to enter “abc.xyz.com” as the value for string input parameters, then service deployment fails due to invalid email address.

  Note: For more information about regex, see regexlib.com.
- **Default value** - If you select, **Specify options**, then enter values for **Label** and **Value** pair fields. Select a record to mark it as default value. You can click the cross mark icon to delete a value record.

- Click **Save**.

The output parameters are displayed at the end along with their names in read-only mode.

Ansible Inventory

An Ansible inventory is a collection of hosts against which jobs may be launched.

After the service with inventory activity is deployed on the Managed services, the inventory is created in the Ansible.

Procedure

1. Select or enter the following values for **Basic information**:

   - **Title** - Enter title of the template activity. It is a mandatory field. It is also the label name that appears in the template activity.
   - **Activity Id** - The Id is automatically generated and is not editable. Click **Copy** to copy the value of the activity Id to your clipboard.
   - **Template name** - It is the name of the template and is not editable.
   - **Template provider** - It is the name of the provider and is not editable.
   - **Version** - Select the template version. A template is always at its default version.

Note: If you select another version and again switch back to the original version, then service parameters gets overwritten by the values from the original template version.

- **Instance name** - Enter the instance name of the template activity.
- **Connection** - Select the Ansible Connection value from the drop-down list.

2. Update the details in **Inventory** and **Hosts** tabs. For more information, see Inventory and Hosts tabs of Ansible Inventory activity.

Note: If the **Editable** flag is set to FALSE, this service does not contribute in the service instance deployment.

Parameters of Ansible Inventory activity

You can set the parameter values for the Ansible inventory from the **Composition** tab > Inventory activity > **Inventory** and **Hosts** tabs.

Procedure

1. Select the activity in the **Composition** tab. The parameters are displayed in the **Activity details** pane.
2. In the **Inventory** tab,
    1. **Inventory Name** - Enter the inventory name suffix. The suffix is used while creating the inventory on the Tower. The inventory is created as ActivityInstanceName_suffix.
    2. The Managed services only supports Default **Organization** at this time.
    3. **Group** - Based on the Ansible playbook, an inventory can target the multiple groups to have groups to host mapping depending on the Ansible playbook functionality.
        1. Click **Add Group** to add the a new group. Enter the name of the group and click ✓. The newly created group is added in the drop-down list.
        2. Click **Delete Group** to delete the selected group from the drop-down list.
    4. **Extra Variables** - These extra variables are the required extra variables by an inventory in the Ansible runbook.
        1. Click **Add Extra Vars** to add extra variables. Click ✓ after entering the **Key** and **Value**.
        2. Click **Delete Extra Vars** to delete the selected extra variable from the drop-down list.
    5. In **Hosts** tab,
        1. Select the **Group** from the drop-down list.
        2. **Host** - The hosts are added for the selected group.
            1. Enter a name of the host or the comma separated list of hosts in the **Host** field. Click **Add Host** to add new host. The newly created host is added in the drop-down list. The list of hosts is displayed at the bottom of the page.
            2. Click **Link Parameter** to link to the existing Managed services input parameter.

Ansible Job Templates

A job template is a definition and set of parameters for running an Ansible job.

The Ansible job templates are discovered on the Managed services after making the Ansible connection. After the service deployment, the job templates are executed on the Ansible.

Procedure

1. Check **Editable** flag if you want to edit these parameters while editing the service instance.
2. Select or enter the following values for **Basic Information**:
    - **Title** - Enter title of the credential activity. It is a mandatory field. It is also the label name that appears in the credential activity.
    - **Activity Id** - The Id is automatically generated and is not editable. Click **Copy** to copy the value of the activity Id to your clipboard.
    - **Template Name** - It is the name of the template and is not editable.
    - **Template Provider** - It is the name of the provider and is not editable.
    - **Version** - Select the template version. A template is always at its default version.

      Note: If you select another version and again switch back to the original version, then service parameters gets overwritten by the values from the original template version.
    - **Instance Name** - Enter the instance name of the template activity.

- **Connection** - There are different ways to enter values for the connection:
    - Steps to map an existing parameter or create a new parameter of type connection:
        - Click **Link Parameter** in the **Activity details** pane to map a new parameter.
        - In the **Link Parameter** dialog box, click **Link** to map an existing parameter.
        - Map an existing parameter or create a new parameter and link it automatically:
            - Link parameters:
                - Select a linking option. You can map parameter values using the **Input Parameters** and **Templates** menu options in the **Link Parameter** dialog box:
                    - **Input Parameters** - The **Input Parameters** menu lists all of the parameters that are available in the **Input Parameters** section of the main **Parameters** tab of a service.
                    - **Templates** - This option lists all parameters of all template activities that are available prior in sequence to this template activity.

                  The values displayed in the **Activity details** pane change based on what you select and which activities you use in your service workflow.

                  Note: For example, if there are two templates A and B placed in sequence in the **Provision Action** of the **Composition** tab, then parameters of both the templates are available in the **Activity Parameters** section of the **Parameters** tab. If you want to link a parameter of template B, then a list of all parameters of the template A and shared parameters are available for association. If your service involves Business Process Manager activities and helm charts, then they are also available for linking. For the very first activity, template A, you can only link input parameters and shared parameters.
                - Expand the parameter menu and hover over the parameters to see a plus icon.
                - Click the plus icon to select it. A green tick mark indicates your selection and also the **Value** is refreshed. You can also use the **Search Parameter** to filter and find a parameter.
                - Click **Save**.
                - You can also click **Remove Link** in the **Link Parameter** dialog box to remove a mapping.
            - Create a new parameter and link it at the same time:
                - In the **Link Parameter** dialog box, click **Create**.
                - Enter a value for the **Parameter Key**.
                - Enter or select values for the new parameter.
                - Enter values for **Read and Write** permission. When you create a service instance, you can make changes to the values that were defined during creation. Enter or select the following values for **Read and Write** permission:
                    - **Parameter Type** - You cannot change the parameter type. It is connection by default.
                    - **Display Name** - The label or display name of the parameter.
                    - **Description** - The description appears as a tooltip when the customer orders this service.
                    - **Cloud Provider** - The cloud provider is pre-selected. You cannot change the value.
                    - **Default Value** - Optionally, enter the value for connection. If you do not provide the value during design time, you must add it during deployment.
                - Click **Save** to create and save the new parameter.
    - You can create a parameter of type string in the **Parameter** tab. In the **Source Code** tab, change its type to connection and add provider value as shown in the following example:

```
"input_parameters": [
{
"name": "Conn",
"label": "Conn",
"Customtype": "string",
"type": "connection",
"provider": "VMware vSphere",
"immutable": false,
"hidden": false,
"required": false,
"secured": false
 }
]
```

Update the details in **Parameters** tab. For more information, see Parameters tab of Ansible Job Template activity.

If a parameter has Read-Write user permission, then you can edit the values to override existing permission and parameter values.

Parameters tab of Ansible Credential activity

You can set the parameter values for Ansible Job Templates from the **Composition** tab > **Job Templates** activity > **Parameters** tab.

Procedure

1. Select the activity in the **Composition** tab. The input parameters are displayed along with their name and value in the **Activity details** pane.
2. **Inventory Name**: Enter the exact name of the inventory used in the Ansible.
3. **Credential Name**: Enter the exact name of the credential used in the Ansible.
4. **Extra Variables**: Enter the extra variables.
5. You can use the **Search Parameters** to filter and search for a parameter.
6. You can edit directly to hardcode the value of a parameter, link the value to a parameter or create a new parameter:
    - Link the parameter value:
        - Click the parameter, then click **Link Parameters** in the **Activity details** pane to map a parameter.
        - Select one of the following options for mapping parameter values: **Note**: Not all options are available for a parameter. It varies based on the activities used in the service.
            - **Input Parameters** - The **Input Parameters** menu option lists all of the available parameters of the main **Parameters** tab of a service.
            - **Shared Parameters Note**: If you link the attribute of a shared parameter, then an input parameter is automatically created. During service deployment, select the dataobject for the datatype. The data attribute value of the selected dataobject is used during provisioning.
            - **Templates** - The **Templates** menu option lists the templates available.
            - **Ansible** - The **Ansible** menu option lists the Ansible job templates.
        - Map values with the parameter:
            - Select an option in the **Link Parameter** dialog box. The values displayed in the **Activity details** pane change based on what you select and which activities you use in your service workflow.
            - Expand the parameter menu and hover over the parameters to see a plus icon.
            - Click the plus icon to select it. A green tick mark indicates your selection and also the **Value** is refreshed. You can also use **Search Parameter** to filter and find a parameter.
    - Create a new parameter:
        - Click **Link Parameter** in the **Activity details** pane.
        - In the **Link Parameter** dialog box, click **Create**. You can create only a new parameter whose type exists in the template activity. For example, a template has string, Boolean, and map parameter types. If you want to create a new parameter of type counter, you must go to the main **Parameters** tab. To work with the main **Parameters** tab, see Working with the Parameters tab.
        - Select **End-User Permissions**.

          The three different types of permissions are as follows:
            - **Invisible** - During parameter creation, enter a value for the **Default Value**. This parameter is not visible while creating an instance of the service. **Note:** For the Invisible parameter, if you enter incorrect data in the default value field, then the Service instance deployment of the content run time fails.
            - **Read-only** - During parameter creation, enter or select the following values:
                - **Parameter Type** - Select a type. It determines the type of parameter. The available parameter types are string, password, list, Boolean, counter, map, connection, and shared parameters.
                - **Display Name** - The label or display name of the parameter.
                - **Description** - The description appears as a tooltip when the customer orders this service.
                - **Default Value** - The default value of the parameter. When you create the service instance, this parameter is visible to you in read-only format.
                - For list data type, you can click **Add New** to add a list of values.
                - For map data type, you can click **Add New** to add a list of **Key Value** pairs. Click cross icon to delete a value record.
                - For password data type, toggle between **Show** and **Hide** to view or hide the password characters.

- **Read and Write** - When you create a service instance, you can make changes to the values that were defined during creation. Enter or select the following values for **Read and Write** permission:
    - **Parameter Type** - Select a type. It determines the type of parameter. The available parameters are string, stringlist, password, list, boolean, counter, and map.
    - **Value Required** - Determines whether a value is required for the parameter.
    - **Specify Options** - For string data type, you can select this option for a drop-down list of string values.
    - **Display Name** - The label or display name of the parameter.
    - **Description** - The description appears as a tooltip when the customer orders this service.
    - **Validation Rule** - The validation rule is a non mandatory field that is available only if you select “string” or “password” value in **Parameter Type** drop-down list. Validation Rule (regex) provides an ability to the service composer to restrict the values of an input parameter. If regex is provided during service composition, then the value of the string input parameter must satisfy the regex criteria during service deployment. For example, (\w+?@\w+?\x2E.+).
    - **Validation Rule Description** - Validation Rule Description (regexdesc) provides information about the validation rule. For example, “Enter a valid email address”. Here, (\w+?@\w+?\x2E.+) is the regular expression for an email address. If you try to enter “abc.xyz.com” as the value for string input parameters, then service deployment fails due to invalid email address.

      Note: For more information about regex, see regexlib.com.
    - **Default Value** - If you select, **Specify Options**, then enter values for **Label** and **Value** pair fields. Select a record to mark it as default value. You can click the cross mark icon to delete a value record.
- Click **Save** to save the changes.

7. The output parameters are displayed at the end along with their names in read-only mode. You can also view the output parameters in the **Source Code** tab at the same level as **`template_params`**.

Pricing for service plans

The cost of a service is represented as a price of the plan selected at the time of deployment and unit of its measure is on a monthly, hourly, and weekly basis.

Currently, pricing in Managed services is at service level and this is expected to be consumed by the third-party applications integrating with Managed services using APIs. Managed services does only syntactic validations. It does not check price entered against Cloud or any other providers.

- List of pricing rules/conditions
- Pricing attributes
- Examples of pricing attribute and their usage in Service Composition Language

List of pricing rules/conditions

- Managed services validations ensure that the parameters used in pricing rules are based on service plan parameters. For example, **`{"gt": ["vm_disk",100,"gb"]},`** here **`vm_disk`** is a service plan parameter with size greater than **`100 gb`** and **`vm_disk`** has to be exposed as a service plan parameter.
- The "any"/"all" condition - "all" specifies that all the conditions must satisfy for pricing. Whereas, "any" indicates that at least one condition must satisfy for pricing.
- Conditions can be defined only on exposed set of plan parameters. You can use only service plan input parameters in the pricing condition.
- List of supported operators supported are "gt", "lt", "ge", "le", "members". The convention to follow for operators are as follows:
    - First element must point to plan parameter. For example, "vm_disk".
    - Seconnd element is the value of that parameter. For example, "40".
    - Third optional element is the unit. For example, "gb".

      For example:

```
"condition":{
                "all":[
                 {"gt":["disk_size",0,"gb"]},
                 {"le":["disk_size",40,"gb"]},
                 {"eq":["instance_type","x2.small"]}
                ]
            },
```

- When defining an "amount", pricing can be a fixed value or it can have dependent plan parameters. You use the plan parameters with ‘incr’ of specified quantity and unit. For example:

  ```
  "amount" : {"value" :49.0},
  ```

  or

  ```
  "amount" : {
              "value" :.99,
              "disk_size":["incr",1,"gb"]
            }
  ```

Pricing attributes

The Managed services captures the following attributes as a part of the Plan definition for the purpose of pricing:

- **`currency`** - It is a 3-character currency code. The default is "USD". For example:

  ```
  "currency": "USD",
  ```

- **`paymentStrategy`** - The values of payment strategy are "MONTHLY", "HOURLY", "YEARLY", "ONCE". For example,

  ```
  "paymentStrategy":"MONTHLY",
  ```

- **`pricingStrategy`** - The values of pricing strategy are as follows:

  - "FIXED" - The price is one time only and fixed for the plan.
  - "RECURRING" - The price is recurring and fixed for the plan.
  - "USAGE" - The price is based on usage as per plan.

  For example:

  ```
  "pricingStrategy" :"RECURRING"
  ```

- **`free`** - It is a Boolean attribute at plan level and its default is ‘false’. For example:

  ```
  "free": false,
  ```

- **`unitofMeasurement`** and **`unitValue`** - The **`unitValue`** is "1" and **`unitofMeasurement`** is "gb" in this example **`"disk_size": ["incr",1,"gb"]`**

Sample plan with pricing

The section provides two examples to illustrate pricing that is defined within plan parameters of a service. For more information about a plan structure, see Plans section in Managing service code structure.

- The following basic pricing sample sets a fixed amount of 99 USD for a virtual server with single CPU, 1 GB RAM, and 50 GB disk on a by-monthly recurring basis.

  ```
  {
   "plans": [
     {
       "name": "Basic",
       "description": " Virtual Server with Single CPU, 1 GB RAM and 50 GB Disk.",
       "metadata": {
         "currency": "USD",
         "prices": {
           "base": {
             "amount": {
               "value": 99.0
             },
             "paymentStrategy": "MONTHLY",
             "pricingStrategy": "RECURRING"
           }
         },
         "bullets": [
           "Fixed pricing",
           "Small virtual machine",
           "Limited storage"
         ],
  ```

```
       "free": false
     }
   }
 ]
}
```

- The following sample plan defines a base price and one additional price for a virtual machine disk. The base price is at a flat cost of 49 USD for a virtual machine disk of size that is between 0 and 40 GB. If the usage goes beyond 40 GB, then additional pricing is considered based on the availability. The condition defines the amount charged for any extra usage of the resource. With the increment of each gb after the base pricing is exhausted, the additional pricing amount of 1 USD is charged.

```
"plans": [{
   "name": "The name of the plan. For example, Amsterdam",
   "description": "The description of the plan. For example, to deploy in Amsterdam",
   "plan_parameters" [
   // In conditions of base and additional pricing section, you can define only those parameters that
are defined here.
     { "name": "vm_disk",
        "label": "VM disk",
        "customtype": "string",
        "type": "string",
        "immutable": true,
        "hidden": true,
        "required": true,
        "secured": false,
        "description": "Virtual Machine disk",
        "default": "10"
       }
      ],
   "metadata": {
       "currency": "The currency used for the transaction. The default is USD.",
       "prices" :{    //This is the main section where you define the pricing.
       "base":{    //  The main pricing is defined in base section.
           "amount" : {"value" : For example, 49.0},
       // The charges for base pricing is mentioned in 'value' element under 'amount'.  As long as the
conditions for base pricing are met, this flat amount is charged. In this example,  49 USD is charged as
long as the parameter "vm_disk" is between 0 to 40 GB.
            "condition":{
        // This conditions section gives a set of rules which determines whether the base pricing is
applicable or not. If the condition is not met, then base pricing becomes invalid and additional pricing
is considered based on its availability.
               "all":[   // It implies the rule set for the price.
                      {"gt":["vm_disk",0,"gb"]},
                      {"le":["vm_disk",40,"gb"]}
                 //["resource name", value, unit of the resource] --->  {"le":["vm_disk",40,"gb"]}

                      //In this example, if the usage is between 0 to 40 GB, then this base pricing is
applicable. If it goes beyond 40 GB, then additional pricing is considered based on the availability.
                    ]
                },
                 "paymentStrategy":"The values could be "MONTHLY", "HOURLY", "YEARLY", "ONCE",
                 "pricingStrategy" : {
                   - "FIXED" : the price is one time only and fixed for the  plan.
                    - "RECURRING":  The price is recurring and fixed for the plan.
                 },
       "additional":[{    //Additional pricing is applicable whenever the base pricing is exhausted.
               "amount" : {    //Defines the amount charged for any extra usage of the resource. --->
"amount" : {    //Defines the amount charged for any extra usage of the resource under additional
pricing.
                  "value" :1,
                  "vm_disk":["incr",1,"gb"]
             //[incr denotes incremental, value, unit]. With the increment of each gb after the base
pricing is exhausted, the additional pricing amount is charged.
               },
               "condition":{
          // This condition section gives a set of rules to determine whether additional pricing is
applicable. If the condition is not met, then this additional pricing becomes invalid. In case another
additional pricing is defined, then it would be considered.
                  "all":[
                     {"gt":["vm_disk",40,"gb"]}
                   ]

               }
            }
            ]
          },
           "bullets" :[ //Main features of pricing.
              "Flexible pricing",
              "Unlimited storage"
           ]
        "free": false,
```

```
    //  It is a boolean attribute at plan level and its default is 'false'. If there is no pricing
attached to a plan, set this is set to true.
                    }
       },
      };
     ]
```

Managing service code structure

The **Source Code** of a service follows Service Composition Language (SCL) V3 version. Use the correct syntax when importing or editing a service.

The Service Composition Language structure is as follows:

```
{
 "service": {
 "specVersion": "v3",
 "catalog_metadata": {
 "name": "",
 "description": "",
 "image": "serviceicon_1.svg",
 "category": "CloudServices",
 "bullets": [
  {
      "title": "Cloud Provider",
      "description": "IBM Bluemix Platform"
    }
 ],
 "providerDisplayName": "IBM",
 "longDescription": "",
 "documentationUrl": "",
 "supportUrl": "",
 "bindable": false,
 "updatable": ""
 "systemTags": true
 },
 "tags": [],
 "datatypes" :      [
   {
      "id" : "",
      "label": "",
      "description": "",
      "datatype": "",
      "default": "",
      "immutable": false
   },
 "actions": [
   {
      "type": "provision",
      "name": "Provision",
      "description": "Default action for deployment of service",
      "input_parameters": [],
      "flow": {
            "conditions": [],
            "templates": [],
            "resthooks": [],
            "notifications": [],
            "sequence": {}
       },
       "output_parameters": []
         }
       ],
       "plans": [
        {
          "name": "Standard",
          "description": "To deploy a Standard plan",
          "actions": [],
          "plan_parameters": []
               "free": It can be true or false. False indicates that it is not free.,
               "metadata": {
                  "currency": "USD",
                  "prices" :{
                    ...                        .
         }
       },
```

For information about the individual sections in the Service Composition Language V3 structure, see the following sections:

- catalog_metadata
- tags
- datatype
- actions
  - input_parameter
  - flow
    - conditions
    - templates
    - resthooks
    - notifications
    - sequence
    - error sequence
- plans

For Service Composition Language samples, see Service samples.

catalog_metadata

```
"catalog_metadata": {
   "name": "The name of the service",
   "description": "A short description of the service",
   "image": "The name of the icon file. For example: serviceicon_1.svg",
   "category": "The category to which the service is assigned. For example: My Category",
   "bullets": [{
        "title": "The name of the first feature",
        "description": "A description for the first feature"
    },
  ]
  "providerDisplayName": "The display name of the provider",
  "longDescription": "A long of description of the service",
  "documentationUrl": "The documentation URL",
  "supportUrl": "The support URL",
  "bindable": true or false,
  "updatable": "This is not supported in this release."
  "systemTags": true or fals. If systemTags flag is set to true, then system generated tags are automatically
added to the tags section of the service instance.
    },
}
```

The information in the `catalog_metadata` section is the information that you provided in the **Overview** tab when you create the service. This information is overwritten if you import a service file in the **Edit Source Code** tab.

The `image` attribute value is the file name of the icon associated to the service. The file name must be `serviceicon_<n>.svg` where `<n>` is a number from 1 to 10. If you specify another file name, the `image` attribute value is changed by default to `serviceicon_1.svg`.

The bullets attribute contains the list of features that you add in the **Overview** tab.

You can add additional fields in `catalog_metadata` to provide additional details about the service in the IBM Cloud Pak for AIOps Catalog.

tags

```
"tags": [
      {
        "name": "The tag name",
        "label": "The display name of the tag",
        "type": "The type of tag.",
        "immutable": Specify if the parameter is immutable. The value must be true or false,
        "hidden": Specify if the parameter is hidden. The value must be true or false,
        "required": Specify if the parameter is required. The value must be true or false,
        "secured": Specify if the parameter is secured. The value must be true or false,
        "description": "The description of the tag",
        "default": "The default value of the tag",
        "customtype": "The custom type of the tag.",
        "permission": "Specify whether the permission. For example, Read-Write",
        }
```

If you select **Specify Options**, then the following additional values are displayed:

```
"options": [
          {
            "label": "The label of the value",
            "value": "The value",
```

```
            "default": Specify if it is a default value. The value must be true or false.
          }
          {
          ...
          ...
          }
        ]
```

datatypes

```
"datatypes" :      [
        {
           "id" : "is an identifier for your datatype instance.",
           "label": "label",
           "description": "Description",
           "datatype": "The Datatype to which it is mapped.",
           "default": "The default DataObject",
           "immutable": "If the definition has `"immutable": false`, then the deployment page shows a drop-
down list of all the DataObjects that are available for the DataType."
        },
]
```

An example for a datatype wherein the default dataobject is defined.

```
"datatypes" :      [
        {
           "id" : "bastionhost1",
           "label": "Bastion Host Configuration",
           "description": "Select Bastion Host Configuration",
           "datatype": "bastionhost",
           "default": "DefaultNoBastionHostRequired",
           "immutable": false
        },
        {
          "id" : "httpproxy",
          "label": "Bastion Host Configuration",
          "description": "Select HTTP Proxy Configuration",
          "datatype": "httpproxy",
          "default": "DefaultNoProxyRequired",
          "immutable": true
        },
    ],
```

If you have only one DataObject associated to a Datatype, then you can directly use it in the interpolation without datatype mapping here. For interpolation syntax, see Using shared parameters in service interpolations.

actions

```
"actions": [
    {
      "type": "provision",
      "name": "Provision",
```

```
      input_parameter [{
        ...
        ...
        }]
        flow {
        ...
        ...
        }
        "output_parameters": []
    }
 ],
 "actions": [
   {
       "type": "bind",
       "name": "Bind",
       input_parameter [{
       ...
       ...
       }]
       flow {
       ...
       ...
       }
       "output_parameters": []
```

```
    }
  ],
```

input_parameter

```
"input_parameter": [{
                "name": "The name of the input parameter. For example: ssh_key",
                "label": "The label to be displayed in the user interface for this parameter",
                "description": "A short description of the parameter. For example, used to access your
server",
                "customtype": "The type of the parameter. For example: String" ,
                "type": "The variable type of customtype. For example: text",
                "immutable": "Specify if the parameter is read-only or not. The value is true for read-only
user permissions and is false for read-write user permission.",
                "hidden": "Specify if the parameter is not displayed in the user interface. The value must be
true or false",
                "required": "Specify if the parameter is required. The value must be true or false",
                "secured": "Specify if the parameter value is shown as ********. The value can be true for
`string` or `password` `customtype` attribute and false for all other attributes.",
                "default": "Default value of the parameter.
                "plans": "


        },
        ...
]
```

If you select **Specify Options**, then the following additional values are displayed:

```
"options": [
          {
            "Default": "Is it a default value of the parameter",
            "value": "The value",
            "label": "The label of the value",
          }
          {
          ...
          ...
          }
        ]
```

flow

conditions

```
"conditions": [
          {
            "decision": {
              "title": "The unique title of the decision.",
              "options": [
                {
                  "case": "0",
                  "sequence": {
                  ...},
                  "case": "1",
                  "sequence":{
                  ...},
                }
              ],
              "id": "The generated id for this component. For example: decisionbbb53a59",
              "description": "The description of the component",
              "mapped_parameter": "The parameter to which this component is mapped. This is a mandatory
parameter"
            }
          }
        ],
```

templates

```
"templates": [{
        "template_name": {
        "title": "The title of the template. For example: 1 Virtual Server with SSH Key",
        "id": "The id of the template component. For example:1virtual6ae83564",
        "template_name": "The name of the template",
        "version:"The version of the template",
        "template_provider: "The provider of the template. For example, IBM",
        "template_content_type": "The content type of the template. For example: prebuilt",
        "instance_name": "The name of the instance associated to this template when deployed",
        "cloud_connection_name": "The name of the cloud connection to be used to deploy the template",
```

```
        "content_runtime": "The runtime content",
        "template_params": [
        {
           "default": "The actual value of the parameter. For read",
           "description": "A description for this template parameter",
           "hidden": "Specify if the parameter is not displayed in the user interface. The value must be true
or false",
           "immutable": "Specify if the parameter is read only. The value must be true or false",
           "label": "The label to be displayed in the user interface for this parameter",
           "name": "The name of the template parameter",
           "required": "Specify if the parameter is required. The value must be true or false",
           "secured": "Specify if the parameter value is shown as asterisk (*). The value must be true or
false",
           "type": "The type of parameter. For example: text",
        },
        {
          ...
        }
      ]
   }
 ]
```

The **type** attribute can have **text** or **list** value. If the **type** value is **list**, in the **default** attribute specify the list of the values that can be assigned to the parameter in the format **["<First value>","<Second value>",...,"<Last value>"]**. For example, **["IBM Cloud","Amazon EC2"]**.

If you want to set a template parameter to the value of a input parameter, specify the **default** attribute value by using the following convention:

```
"default": "${input_parameters.<service_parameter_name>}
```

For example, you have the following input parameters:

```
"input_parameters": [{
   "name": "datacenter",
   "label": "datacenter",
   "description": "Target vSphere datacenter for VM creation",
   "type": "list",
   "default": ["vSphere datacenter 1","vSphere datacenter 2"],
   "hidden": "false",
   "immutable": "false",
   "required": "true",
   "secured": "false",
   "validation": "",
   "dependent": "",
   "dependent_attribute": "",
   "return_type": "text"
},
```

You can define the following template parameter to set its value to the related input parameter value that you select when ordering the service:

```
"template_params": [
                {
                "name": "datacenter",
                "label": "datacenter",
                "description": "Target vSphere datacenter for VM creation",
                "type": "text",
                "default": "${input_parameters.datacenter},
                "hidden": "true",
                "immutable": "false",
                "required": "true",
                "secured": "false",
                "validation": "",
                },
```

In the previous example, note that the **hidden** attribute value in the template parameter must be set to **true** to allow the template parameter value to be replaced with the input parameter value when ordering the service.

If you are using a Business Process Manager template, then the following parameters are added to the template:

```
"async_req": true  //flag to determine whether it is a long running template.
"long_running_delay": "<time in minutes>", //Delay after every poll. Value must be in minutes.
"long_running_timeout": "<time in hour>" //Total time for which polling must be run. The value must be in
hours.
```

Example of a Business Process Manager template:

```
"templates": [
  {
    "SampleLongRunningProcess": {
      "title": "SampleLongRunningProcess_9de34f",
      "template_name": "SampleLongRunningProcess_25.00746596-7ced-4f98-b382-294e2f702e95_2063.5af7b495-770a-
```

```
4408-a15b-76ab6f3e890e",
      "version": "1.0"
      "id": "sample10139690a6",
      "template_type": "BPM",
      "template_content_type": "BPM",
      "template_provider": "BPM",
      "instance_name": "",
      "cloud_connection_name": "BPM Connection",
      "template_data_objects": {},
      "async_req": true,
      "long_running_delay": "5",
      "long_running_timeout": "24",
      "template_params": {
          "input1": ""
      },
      ...
      ]
```

Here, in 24 hours, poll happens at a frequency of 5 minutes and in total 288 times. if the Timeout is 24 and delay is 5, then poll happens 24*60/5 = 288 times.

resthooks

```
"resthooks": [{
        "resthook": {
        "title": "The title of the resthook",
        "id": "The unique id of the resthook",
        "header": {
                "accept": "
                "Content-Type: "application/json",
                The header of the resthook in JSON string format"
                },
                "url": "The URL of the external REST server",
                "method": "The method to poll the external REST server",
                "payload" : "The request payload if you specified a POST method"
                "timeout": 120,
        }
        "on_destroy":
                {
                "headers": {
                  "accept": "application/json",
                  "Content-Type": "application/json"
                },
                "auth": {
                  "type": "basic",
                  "username": "administrator@vsphere.local",
                  "password": "^(RGFuYzFuZyE=)"
                },
                "url": "https://example.com/create",
                "method": "POST",
                "payload": {
                  "id": "1"
                }
                "timeout": 120,
            }
      }
    ],
      ...
]
```

The service REST hooks are used to poll an external REST server.

Note: The **timeout** attribute suggests the maximum time of an URL connection. The default value for **timeout** is 120 seconds. As the **timeout** attribute is not available by default in Rest Hooks, you must add the timeout attribute and configure it as per your requirement. The resthook time out occurs either when the network timeout occurs or when the **timeout** value is reached.

notifications

```
"notifications": [{
        emailnotification":{
                "title": "The title of the email notification component",
                "type": "The type of the service notification. For example: email",
                "id": "The identifier of the email notification component",
                "sender": "The sender address",
                "receiverList": "The list of receiver email addresses",
                "subject": "The subject of the email",
                "body": "The notification body",

      },
```

```
      ...
]
```

Before using service notifications, you must configure a connection to the SMTP server. For more information, see Configuring a connection for notifications.

sequence

It defines the sequence of activities to deploy the service instance.

For example:

```
"sequence": {
            "0": "ibmchartf76699ee",
            "1": "emailnot43bfbb6f"
          }
```

error sequence

It defines the sequence of activities to deploy a error sequence defined in the **Error flow** section of the **Composition tab**.

For example:

```
"error_sequence":{
        "0": servicen247273d"
        }
```

plans

You can have multiple plans with or without pricing.

Example plan without pricing details:

```
"plans": [{
      "name": "The name of the plan. For example, Amsterdam",
      "description": "The description of the plan. For example, to deploy in Amsterdam",
      "actions": [],
      "plan_parameters" [{
            "name": "vm_cpu",
            "label": "Server CPU",
            "customtype": "string",
            "type": "string",
            "immutable": true,
            "hidden": true,
            "required": true,
            "secured": false,
            "description": "Virtual Machine CPU",
            "default": "1"
          }
        ]
            "free": true
          },
      };
     ]
```

Where

- `plan_parameters[]` - If your service has some input parameters in the provision action that you want to override at the time of deployment, use `plan_parameters`.
- `free` - Signifies whether pricing is associated to the plan or not. If it is associated, then the value is set to false.

For an example of a plan with pricing, see Sample pricing.

Supported interpolation expressions

Managed services service components have the capability to take an input parameter from another service component, hence links to all composer supported interpolation expressions are also included here.

The composer enables you to connect input parameters to service parameters or output parameters of other activities. These parameters are determined by composer supported expressions.

- Input and output parameter formats of all activities
- Inbuilt input and output formats of all activities

- Detailed information with examples for all activities
- Troubleshooting

Input and output parameter formats of all activities

Input and output parameter formats of the different activities are tabulated as follows:

Format	Example	Refers to
`${templates.<ID>.output.<PARAM_NAME>}`	`${templates.tomcat019453ac.output.TomcatNode01_ip}`	Output Parameter of a template. It is applicable only for terraform template as Helm does not support output parameters.
`${templates.<ID>.<PARAM_NAME>}` or `${templates.<ID>.input.<PARAM_NAME>}`	`${templates.tomcat019453ac.TomcatNode01-mgmt-network-public}`	Input Parameter of a template
`${input_parameters.<PARAM NAME>}`	${input_parameters.Conn}	Dynamic parameters for template activity. Here, `<PARAM NAME>` can be a variable of type `connection` or `sharedparameter` that you need to configure as input parameter during service definition. For more information about shared parameter, see Shared parameters in Templates topic. For more information about connection, see the Different ways of adding values for the cloud connection in Templates topic. For the behavior of these dynamic parameters at runtime, see Creating a service instance.
`${resthooks.<ID>.output.<PARAM_NAME>}`	`${resthooks.resthookf342a67e.output.body}`	Output parameter of a Rest Hook.
`${resthooks.<ID>.input}`		Payload of the Rest Hook.
`${resthooks.<ID>.output}`		Response or output or body parameter of Rest Hook.
`${notifications.<ID>.output.<PARAM_NAME>}`		Output parameter of notification.

Inbuilt input and output formats of all activities

The inbuilt input parameters for activities are tabulated as follows:

Format	Refers to
`${templates.<id>.instance_name}`	Name of the template instance
`${templates.<id>.template_name}`	Name of the template
`${templates.<id>.template_version}`	Version of the template
`${templates.<id>.title}`	Title of the template
`${templates.<id>.stack_id}`	Get the stack id of an activity
`${resthooks.<id>.url}`	URL of the Rest Hook.
`${resthooks.<id>.method}`	Method
`${resthooks.<id>.headers.<name>}`	Name of the header
`${resthooks.<id>.payload}`	Payload
`${resthooks.<id>.authentication.type}`	Authentication type
`${resthooks.<id>.authentication.username}`	User name
`${resthooks.<id>.headers.Content-Type}`	Content type headers
`${resthoooks.<id>.title}`	Title of Rest Hook
`${notifications.<id>.to}`	Receiver email address.
`${notifications.<id>.cc}`	Copy to email addresses.
`${notifications.<id>.bcc}`	Bcc to email addresses.
`${notifications.<id>.sender_name}`	Email id of the sender.
`${notifications.<id>.subject}`	Subject of the notification.
`${notifications.<id>.message}`	Notification message.
`${notifications.<id>.title}`	Title of the notification.
`${decisions.id.selection}`	Retrieve data about the decision that got selected at the time of deployment.

The inbuilt output parameters for the different activities are tabulated as follows:

Format	Refers to
`${templates.<id>.action}`: `CREATE` / `DELETE`	Create or delete action
`${templates.<id>.status}` : `SUCCESS` / `IN_PROGRESS` / `FAILED`	The status of a template. It can be success, in progress, and failed
`${templates.<id>.status_message}`	Additional message for the status
`${templates.<id>.stack_message}`	Template execution stack logs (from IAAS)
`${templates.<id>.stack_error_message}`	Template error message (from IAAS)
`${templates.<id>.created_at}`	Time of creation of the template. (from IAAS)
`${notifications.<id>.status}` : SUCCESS / FAILED	Status of the email notification.
`${notifications.<id>.status_message}`	Additional message for the status.

Detailed information with examples for all activities

The following table has links to input and output parameter mapping details of all activities along with their examples:

Service component / General	Input and output parameter mapping
Templates and helm charts	Input output mapping for templates and helm charts
Integration > Resthooks	Input and output parameter mapping for resthooks
Flow components > Decision	Parameter mapping for decision
Notifications > Email notification	Input and output parameter mapping for email notification
Service instance level parameters	Service instance level parameters
Interpolation parameters of Error flow	Service instance level parameters in Error flow
Shared parameters in service interpolations	Using shared parameters in service interpolations

Troubleshooting

If a service references a string literal, such as `${<text>}` in the parameter value of an activity, then use `$${<text>}` as a string to escape it in the service definition. For example, you have a helm chart that contains `${path.config/prospectors.d*.yml}` in **Filebeat.config.modules.path**. When you create a service to deploy this helm chart, the following error occurs:

```
Error: Invalid resource : ${path.config} (hint: 'path' not found)
{"error":{"statusCode":500,"message":"Internal Server Error"}}
```

To resolve this error, change `${path.config}` to `$${path.config}`.

For more troubleshooting information, see Troubleshooting Service Composer or Portal issues.

Input output mapping for templates and Helm charts

Mapping procedure in the Managed services user interface and their input output formats.

- Template / Helm input parameters
- Template / Helm output parameters
- For service instance level parameter details and service instance level parameters in Error flow, see Service instance level parameters.

Template / Helm chart input parameters

Representing input parameters and attributes of an activity within another activity.

Note: The `templates` word in the syntax is used for both template and helm activities.

If your source parameter is defined in the input parameters section of an activity, then use the following format to call its value within another activity:

```
${activity.id.paramname}
```

Examples:

- For example, `Apache Tomcat on a single virtual machine` template activity has `TomcatNode01-mgmt-network-public` parameter defined in the **INPUT PARAMETERS** section. Use the following format to call its value within another activity:

  ```
  ${templates.tomcat019453ac.TomcatNode01-mgmt-network-public}
  ```

  or

`${templates.tomcat019453ac.input.TomcatNode01-mgmt-network-public}`

Where **tomcat019453ac** is the activity Id.

- For example, mapping the output of **IPAM** template as an input to **Oracle Database 12c Enterprise Edition on a single virtual machine template**:

  ```
  OracleDBNode01_dns_suffixes": "${templates.ipam1467a9b92b6c.output.ipam_result_domain_list},
  OracleDBNode01_domain": "${templates.ipam1467a9b92b6c.output.ipam_result_domain_str},
  OracleDBNode01-name": "${templates.ipam1467a9b92b6c.output.ipam_result_host},
  OracleDBNode01_ipv4_address": "${templates.ipam1467a9b92b6c.output.ipam_result_ip_address},
  ```

  Where

  - **OracleDBNode01** is the ID of**Oracle Database 12c Enterprise Edition on a single virtual machine** template.
  - **ipam1467a9b92b6c** is the ID of **IPAM** template.

  The sample snippet of Service Composition Language wherein the output of one template activity is mapped as input into another template activity:

  ```
  "templates": [
      {
        "IPAM": {
          "title": "IPAM_b76b9c",
          "template_name": "IPAM",
          "version": "v1.0.0",
          "id": "ipam1467a9b92b6c",
          "template_type": "Terraform",
          "template_content_type": "Other",
          "template_provider": "Other",
          "instance_name": "",
          "cloud_connection_name": "Other",
          "template_data_objects": {},
          "template_params": {
            "ipam_ip_address": "9.9.9.9",
            "ipam_result_domain": "domain",
            "ipam_result_host": "hostname",
            "ipam_result_ip_address": "ipaddress"
          },
          "outputs": [
            "ipam_result_domain_list",
            "ipam_result_domain_str",
            "ipam_result_host",
            "ipam_result_ip_address"
          ]
        }
      },
      {
        "Oracle Database 12c Enterprise Edition on a single virtual machine": {
          "title": "Oracle Database 12c Enterprise Edition on a single virtual machine_6ec8ab",
          "template_name": "Oracle Database 12c Enterprise Edition on a single virtual machine -
  VMware vSphere",
          "version": "2.1",
          "id": "oracledaff0e8430",
          "template_type": "Terraform",
          "template_content_type": "VMware vSphere",
          "template_provider": "VMware vSphere",
          "instance_name": "",
          "cloud_connection_name": "VMware",
          "template_data_objects": {
            "bastionhost": "DefaultNoBastionHostRequired",
            "httpproxy": "DefaultNoProxyRequired",
            "advanced_content_runtime_chef": "VMware"
          },
          "template_params": {
            "OracleDBNode01_dns_servers": [
              "9.5.46.5",
              "9.5.48.5"
            ],
            "OracleDBNode01_dns_suffixes":
  "${templates.ipam1467a9b92b6c.output.ipam_result_domain_list}",
            "OracleDBNode01_domain": "${templates.ipam1467a9b92b6c.output.ipam_result_domain_str}",
            "OracleDBNode01-image": "terraform/managed_services/content/RHEL7-content",
            "OracleDBNode01-os_admin_user": "root",
            "OracleDBNode01-os_password": "aaaaaa",
            "OracleDBNode01_oracledb_version": "v12c",
            "OracleDBNode01_oracledb_port": "1521",
            "OracleDBNode01_oracledb_release_patchset": "0.0.0.0.0",
            "OracleDBNode01_oracledb_SID": "ORCL",
            "OracleDBNode01_root_disk_size": "250",
  ```

```
          "OracleDBNode01_oracledb_security_sys_pw": "aaaaaa",
          "OracleDBNode01_oracledb_security_system_pw": "aaaaaa",
          "OracleDBNode01-name": "${templates.ipam1467a9b92b6c.output.ipam_result_host}",
          "OracleDBNode01_root_disk_datastore": "datastore1",
          "OracleDBNode01_ipv4_gateway": "9.9.9.9",
          "OracleDBNode01_ipv4_address":
"${templates.ipam1467a9b92b6c.output.ipam_result_ip_address}",
          "OracleDBNode01_memory": 8192,
          "OracleDBNode01_ipv4_prefix_length": "24",
          "OracleDBNode01_number_of_vcpu": 2,
          "OracleDBNode01_network_interface_label": "VM Network",
          "OracleDBNode01_cluster": "icm-vmware-nsx-2",
          "OracleDBNode01_datacenter": "NSX Manage To",
          "OracleDBNode01_folder": "terraform/test",
          "OracleDBNode01_adapter_type": "vmxnet3",
          "OracleDBNode01_resource_pool": "cam",
          "user_public_ssh_key": "ssh-rsa AAAABCCCCCCaaaaaaaLDCU3jpchru5Q== abc@in.ibm.com"
        },
        "outputs": [
          "OracleDBNode01_ip",
          "OracleDBNode01_name",
          "OracleDBNode01_roles",
          "stack_id"
        ]
      }
    }
```

- The following input parameters are common for all template/Helm activities:

  Format:

  ```
  ${templates.<id>.<attribute name>}
  ```

  ```
  ${templates.<id>.instance_name} \\ Name of the instance
  ${templates.<id>.template_name} \\ Name of the template activity
  ${templates.<id>.template_version} \\ Version of a template activity
  ${templates.<id>.title} \\ Title of a template activity
  ${templates.<id>.stack_id} \\ Stack id of an activity
  ```

Template / Helm output parameters

Representing output parameters and attributes of a template / helm activity within another activity.

Note: The `templates` word in the syntax is used for both template and helm activities.

- Use the following format for a specific template/helm activity:

  If you want to map your parameter to the output parameters of another template/helm activity, then prefix the word "output" when you reference it in the value field.

  ```
  ${activity.id.output.paramname}
  ```

  The `param_name` is the name of the parameter that is returned by a particular template activity or helm activity. It depends on the template definition for terraform templates, and for helm chart, it depends on the parameters defined in IBM Cloud Pak for AIOps.

  For example, `Apache Tomcat on a single virtual machine` template activity has `TomcatNode01_ip` defined in the **OUTPUT PARAMETERS** section. Use the following format to call its value within another activity:

  ```
  ${templates.tomcat019453ac.output.TomcatNode01_ip}
  ```

  Where, `tomcat019453ac` is the template activity Id and `TomcatNode01_ip` is the parameter.

- The following output parameters are common for all template/helm activities:

  Format:

  ```
  ${templates.<id>.<attribute name>}
  ```

  ```
  ${templates.<id>.action} : **`CREATE`** / **`DELETE`**
  ${templates.<id>.status} : **`SUCCESS`** / **`IN_PROGRESS`** / **`FAILED`**
  ${templates.<id>.status_message} :  additional message for the status
  ${templates.<id>.stack_message} : template execution stack logs (from IAAS)
  ${templates.<id>.stack_error_message} : template error message (from IAAS)
  ${templates.<id>.created_at} : template created at. (from IAAS)
  ```

Input and output parameter mapping for REST Hook

The mapping of input and output parameters for REST Hook and their formats.

- REST Hook input parameters
- REST Hook output parameters
- Example scenarios to call the input and output parameters
- For service instance level parameter details and service instance level parameters in Error flow, see Service instance level parameters.

REST Hook input parameters

The following input parameters are common to all REST Hook activities:

Format:

```
${resthooks.<id>.<attribute name>}
```

```
${resthooks.<id>.url}
${resthooks.<id>.method}
${resthooks.<id>.headers.<name>}
${resthooks.<id>.payload}
${resthooks.<id>.authentication.type}
${resthooks.<id>.authentication.username}
${resthooks.<id>.headers.Content-Type}
${resthoooks.<id>.title}
```

```
${resthooks.<ID>.<input>}
```

REST Hook output parameters

Representing output parameters and attributes of a REST Hook activity within another activity.

- Response or output or body parameter of REST Hook:

  ```
  ${resthooks.<ID>.output.body}
  ```

  To get the ID of an activity, see the **Source Code** tab. The `param_name` is the name of the parameter that is returned by a particular REST Hook activity.

  For example, the output of REST Hook `resthook_f4c71a` is referenced in email notification `emailnotification_c5ee13`:

  ```
  "resthooks": [
          {
            "resthook": {
              "title": "resthook_f4c71a",
              "id": "resthookf342a67e",
              "headers": {
                "accept": "application/json",
                "Content-Type": "application/json"
              },
              "url": "http://9.5.28.173:3000/posts/test_on_error_flow",
              "method": "GET"
            }
          }
        ],
        "notifications": [
          {
            "emailnotification": {
              "title": "emailnotification_c5ee13",
              "type": "email",
              "receiverList": "${input_parameters.to}",
              "id": "emailnot745071d6",
              "sender": "CAMadmin",
              "copyToList": "",
              "subject": "Rest server is accessible",
              "body": "${resthooks.resthookf342a67e.output.body}"
  }
  ```

- The following output parameters are common to all REST Hook activities:

  Format:

  ```
  ${resthooks.<id>.<attribute name>}
  ```

  ```
  ${resthooks.<id>.status} : SUCCESS / FAILED
  ${resthooks.<id>.body} : response body, JSON object
  ${resthooks.<id>.status_code} :http status code. For example, 200 OK
  ${resthooks.<id>.status_message} : http status. For example, 200 OK
  ```

Example scenarios to call the input and output parameters

The following example scenarios illustrate the various conventions that you must use to call the input and output parameters:

- Example 1

```
{
  "ip" : "9.9.9.9",
  "hostname" : "vm1"
 }
```

Syntax:

```
${resthooks.id.body.ip}
```

Result is "9.9.9.9

- Example 2

```
{
  "data" : {
  "ip" :  "9.9.9.9",
   "hostname" : "vm1"
   }
 }
```

Syntax:

```
${resthooks.id.body.data.ip}
```

Result is "9.9.9.9

- Example 3

```
{
  "zones" : [
     {
       "name" : "east",
       "percentage" : "25"
      },
      {
        "name" : "west",
        "percentage" : "70"
  }
   ]
  }
```

Syntax:

```
${resthooks.id.body.zones[name=east].percentage}
```

or

```
${resthooks.id.body.zones[0].percentage}
```

Result: 25

Note: In the following example, `dns_servers` is a variable of list type that is an output from a REST Hook activity:

```
{
  "id": "8",
  "domain": "domain.kstart.com",
  "hostname": "hostname-cam-local",
  "ipaddress": "9.9.9.9",
  "domain_suffix": [
  "domain.kstart.com",
  "domain.kstart.com"
   ],
  "dns_servers": [
  "9.9.9.0",
  "9.9.9.1"
   ]
 }
```

The value for the list type is an element of a list and is not a list by itself, so do not enter the following value in the **Value** field of **Edit Parameter** > **Update** tab for a list parameter type:

```
"vm_1_dns_servers": ["${resthooks.resthookc15774af.output.body.dns_servers}"
        ],
```

If you enter the reference JSON value in the **Value** field, then the following error message is displayed:

```
Could not parse value for param [vm_1_dns_servers] of type [list] with value : ["["9.9.9.0","9.9.9.1"]"]
```

Instead, go to the **Source** tab to reference a value of type list/map.

Parameter mapping for decision

Parameter mapping for decision activity.

If you want to retrieve data about the decision that got selected at the time of deployment, use the expression `${decisions.id.selection}`. For example `${decisions.decision6d43ee5f.selection}`.

OR (||) Parameter mapping in decision

The cases defined within a decision activity have different output string expressions and the actual output is decided during runtime. As you are not aware of the case that gets selected beforehand, use a OR (||) operator as a delimiter between the output of the cases. The OR (||) operator returns a single first non-nil value. The non-selected parameters always have a nil value and are not considered for further processing.

For example, there is a decision activity that has two cases:

- The output of the first case is `templates.template4ce5b4be.user`
- The output of the second case is `templates.template105c1603.user`

The OR (||) operator expression for this example is as follows:

```
${templates.template4ce5b4be.output.user || templates.template105c1603.output.user}
```

If the first case is selected during runtime, then `templates.template4ce5b4be.user` is used for further processing in the workflow as it is the first non-nil value. The `templates.template105c1603.user` has a nil value and is not considered.

If an activity parameter is from inside a decision block and the decision block is not selected, then the parameter (output or input parameter) evaluates to an empty value.

If an activity parameter is from a selected decision block having an empty-value string, then the next parameter having a value string is considered.

If you use interpolation in the error flow around a non executed decision from the primary flow, the resultant value is an empty string.

Input and output parameter mapping for email notification

The output and input parameters for email notification.

Email notification output parameters:

- The following output parameters are common to all email notification activities:

  ```
  ${notifications.<id>.<attribute name>}
  ${notifications.<id>.status} : SUCCESS / FAILED
  ${notifications.<id>.status_message} : additional message for the status
  ```

  Note: The `status` is the send request for the email that was successfully accepted by the SMTP server.

Email notification input parameters:

- Use the following format for input parameters of notification activity:

  ```
  ${notifications.<id>.to}
  ${notifications.<id>.cc}
  ${notifications.<id>.bcc}
  ${notifications.<id>.sender_name}
  ${notifications.<id>.subject}
  ${notifications.<id>.message}
  ${notifications.<id>.title}
  ```

Service instance level parameters

The conventions to be followed for service instance level input and output parameters.

The `svc_instance` represents a JSON request object that is used to initiate a "provision" or a "bind" action instance.

- `${svc_instance.name}` - Name of the "provision" or a "bind" action instance that is entered in the deployment page.
- `${svc_instance.instance_parameters}` - The action parameter JSON object with key/values as entered in the deployment page. The Keys depend on the service definition. For example:

  ```
  {
    "host" : "some-name",
    "region" : "us-west"
  }
  ```

- `${svc_instance.instance_parameters.host}` - The particular action parameter key value. Using a subquery to drill down to a particular value.
- `${svc_instance.service}` - The service definition JSON object as available in the service composer source tab.

  Note: The `${svc_instance.service}` interpolation interpolates the whole source of the service. The service source might contain other interpolations, the Service composer tries to resolve these nested interpolations, which might sometimes cause undesired errors. So it is not recommended to use `${svc_instance.service}`, if used, should be used with caution.

- `${svc_instance.service.actions[name=provision].input_parameters[name=HybridCloud].options[label=AWS].value}` - Using a subquery to drill down to a particular value in the service definition. It gets the value of one of the input parameter options. In this example, it is "HybridCloud", "AWS".
- `${svc_instance.service.catalog_metadata.category}` - Using subquery to drill down to a particular value in the service definition. In this example, it gets the category of the service, "CloudServices".
- `${svc_instance.owner}` - Owner of the service. For example, "admin".
- `${svc_instance.instance_plan}` - Plan name in use for the current instance. For example, "Standard".
- `${svc_instance.tags}` - Tags JSON object with key/value pairs of tag name and values as entered in the Deployment page. Tag key names depend on the service definition. For example:

  ```
  {
   "dept" : "FIN"
   "org" : "AAA"
  }
  ```

- `${svc_instance.service_action_name}` - Current action has been performed. Name maps to `action.name` in the service definition. For example, "Provision".
- `${svc_instance.output_parameters.<parameterName>}` - Key/value pairs of output parameters received from 'Provision' action. Keys are defined in service definition. These output parameters are available in 'Bind' action. For example:

  ```
  {
    "host" : "test01.ibm.com",
    "ipaddress: "9.9.9.9"
  }
  ```

  You can drill down to a particular key in the output parameters as in the following example:

  ```
  `${svc_instance.output_parameters.ipaddress}`
  ```

  This returns the IP address "9.9.9.9"

  For more information on how to define output parameters, see Creating a service.

- `${svc_instance.namespace}` - The namespace value for a deployed instance.

Service instance level parameters in Error flow

- In addition to the listed service instance level parameters, the error flow section of the **Composition** tab supports these interpolations:
  - `${svc_instance.error.message}` - Error message for failure
  - `${svc_instance.error.activity_id}` - Activity id of the failed activity
  - `${svc_instance.error.activity_type}` - Type of failed activity
  - `${svc_instance.error.created_at}` - Creation time of the failed activity

    - `${svc_instance.primaryflow.decisions.<decision_id>.selection}` - Retrieve data about the decision that got selected at the time of deployment. For example, `${svc_instance.primaryflow.decisions.decision6d43ee5f.selection}`.
- Interpolation of primary flow activity parameters in Error flow:

  The interpolation expression for primary flow activity parameters in Error flow is as follows:

  `${svc_instance.primaryflow.<activity>.<activity_id>.<attribute name>}`

  Where, `activity` can be REST Hook, templates, or notifications.

  The 'svc_instance.primaryflow.' prefix can be used with any parameter of REST Hook/template/notification activity from the primary flow. If an `${svc_instance.primaryflow.xxx.xxx.xxx}` expression does not exist, then an empty string value is used.

  Note: The `${svc_instance.primaryflow.xxx.xxx.xxx}` expression works only from within an error flow section.

- Example interpolation expressions for the different activities:
    - REST Hook:
        - `${svc_instance.primaryflow.resthooks.resthook8a5f70ec.payload}` - REST Hook payload
        - `${svc_instance.primaryflow.resthooks.resthook8a5f70ec.status}` - REST Hook status
        - `${svc_instance.primaryflow.resthooks.resthook8a5f70ec.headers.Content-Type}` - REST Hook Headers
        - `${svc_instance.primaryflow.resthooks.resthook8a5f70ec.authentication.type}` - REST Hook authentication mechanism
        - `${svc_instance.primaryflow.resthooks.resthook8a5f70ec.authentication.username}` - The username for REST Hook authentication management system.
        - `${svc_instance.primaryflow.resthooks.resthook8a5f70ec.title}` - The title of the REST Hook.

      For REST Hook related input and output interpolation expressions, see Input and output parameter mapping for REST Hook.
    - Templates:
        - `${svc_instance.primaryflow.templates.testtemp53db922a.status}` - Template status
        - `${svc_instance.primaryflow.templates.testtemp53db922a.output.selected-region}` - Template Output: Selected Region
        - `${svc_instance.primaryflow.templates.testtemp53db922a.template_name}` - Template Name
        - `${svc_instance.primaryflow.templates.testtemp53db922a.input.sleep}` - Template: Input Param Sleep

      For template-related input and output interpolation expressions, see Input output mapping for templates and Helm charts.
    - Notification Status:
        - `${svc_instance.primaryflow.notifications.emailnotbf32ca47.status}`
        - `${svc_instance.primaryflow.notifications.emailnotbf32ca47.sender_name}`
        - `${svc_instance.primaryflow.notifications.emailnotbf32ca47.title}`

      For notification-related input and output interpolation expressions, see Input and output parameter mapping for email notification.

Manually using shared parameters in service interpolations

Shared parameters can be manually linked and used in service interpolations.

For linking shared parameters from the user interface, see Link parameter section in Managing an existing activity parameter and Read-write End-User Permissions in Parameters tab of Helm charts / Templates activity section.

The shared parameter interpolation syntax that you can use to manually link:

`${datatypes.<DATATYPE_ID>.<attribute name>}`

Where,

- `DATATYPE_ID` is an identifier of Data Type to Data Object mapping that is defined in the service file.
- `attribute name` is the attribute name from the Data Type definition.

For example:

```
"actions": [
      {
        "type": "provision",
        "name": "Provision",
        "description": "Default action for deployment of service",
        "input_parameters": [
        {
            "name": "bastionhost1",
```

```
            "label": "Bastion Host Configuration",
            "customtype": "sharedparameter",
            "type": "sharedparameter",
            "immutable": false,
            "hidden": false,
            "required": true,
            "secured": false,
            "description": "Bastion Host Configuration",
            "default": "DefaultNoBastionHostRequired",
            "datatype": "bastionhost"
          }
```

For more information on the attributes and a detailed example, see Data Types in Service Composition Language.

In this example, **the DATATYPE_ID** is "bastionhost1" and it is mapped to **bastionhost** Data Type.

If there are multiple DataObjects created for a DataType, then a Data Object must be identified as a default object. In this example, the default DataObject is **DefaultNoBastionHostRequired**. In the following interpolation syntax, you override the default Data Object and make use of **bastion_host** Data Object:

```
${datatypes.bastionhost1.bastion_host}
```

The element **datatypes** is included in SCL V3 to support Shared Parameters in service definition, that is, DataObject to Datatype mapping. For more information, see datatypes section of Editing a service file topic.

Note: Only the Data Types that are created as string are supported in service context.

Service validations

Whenever you work with your service in the Managed services user interface, service validation is performed to raise early warnings and errors. These preemptive warnings and errors are displayed in the navigator panel. They enable you to create a service that is free of any errors or warnings.

User interface validations are triggered when you make changes to the service and API validations are triggered when you click **Save** or **Publish** buttons. Click the validation in the notification panel to find out more information about the context of error messages and warnings. In addition, in the composer canvas, a small icon (error and/or warning) is displayed against the related activity that has a validation problem.

Service validation approaches are as follows:

- User interface validation - It occurs as soon as you edit a new service or already existing service.
- API validation - It occurs as soon as you save the service. The API validation errors are no longer displayed if you try to modify the service or move away from the composer tab. Some validation results in error, while others result in warning. API can return either errors or warning at a time, so if a service has both errors and warnings, then the user interface does not show both at the same time. You must first fix the errors and click save to see the warnings. For a list of probable API validation messages, see API Validations.
- Publish time validation - It occurs as soon as you publish a service.

The validation messages are categorized in the following manner for the first two approaches:

- Warnings and problems that do not prevent you from saving a service.
- Errors and critical problems that prevents you from saving a service

List of all probable validation messages that occur when you create, save, or publish a service:

- Validations in composer
- Warnings in composer
- Errors on Save
- API Validations

Validations in composer

- These warnings check the source code for various false conditions, such as empty values, mandatory values for decision, Rest Hook, email notification, and templates .
- The source code is validated whenever there is a change anywhere in the service.
- The JSON syntax errors that are editor errors are also taken care, but the message does not show the exact point of failure.
- REST API errors and validation framework errors at the time of save.

Warnings in composer

{: #warnings]

- When service 'catalog_metadata.name' key is missing - Error is thrown from API
- When name of the service 'catalog_metadata.name' value is missing - Error is thrown from API
- Template key 'catalog_metadata' is missing from the SCL - Error is thrown from API
- If 'longDescription' in overview has length more than 400 chars.
- When service 'description' has length more than 150 characters
- When feature 'bullet's description' has length more than 150 characters
- When there is decision key missing - Error is thrown from API
- When title of the decision, that is, `decision.title` value is missing.
- When `decision.mapped_parameter` value is incorrect, that is, it does have interpolation syntax error.
- When `decision.mapped_parameter` value is missing from SCL.
- If there is no condition array in the SCL or "conditions" + " key is missing - Error is thrown from API
- If notifications array "notifications" + " key is missing" - Error is thrown from API
- Parameters of email notification like title and receiver list key or value is missing.
- The Rest Hook array or key is missing. - Error is thrown from API
- The Rest Hook URL is missing for create/on_delete.
- The Rest Hook URL value is missing from the SCL for create/on delete.
- URL is missing from both the create or delete definition of resthook.
- Templates array or 'templates' key is missing. - Error is thrown from API.
- Templates title key and value is missing.
- Templates cloud connection key or value is missing.
- The `template_params` are there but name key or value is missing from the template_params.
- Validation for hidden parameters for template_params.
- Template is not a valid JSON - Error is thrown from API.
- Parsing errors for SCL JSON - Error is thrown from API.

Errors on Save:

- REST error responses from various rest calls.
- Validation for using same name for the service in a namespace.
- Empty JSON values for different data structure object like json, array, map etc,
- Mandatory parameters missing from the various flow components - REST Hook, Decision, Email, Templates etc,
- Syntax errors.
- Publish time validations.

API Validations

Validations at the time of service creation

- Check whether service name is present or not - error
- Duplicating a service with same name not possible - error
- Tag present is invalid - error and if it is empty, then assign it to cloud services - warning (at the time of save and edit)
- Image provided is valid or not, if image is invalid replace it with a valid one - warning (at the time of save and edit)
- Check the overall json structure in source code, which will include checks for the presence of service attribute, spec_version, catalog_metadata, name, actions attribute of type array, flow attribute of type json, conditions, Rest Hook, notifications, plans and templates section of type array, sequence attribute of type json) - error
- Checking if sequence object has valid ids and is not empty - error
- Check if plan name is unique. - error
- Check if service name is unique. - error
- Check if parameters present in the source code are of correct Managed services variable format (check if name, hidden, required, immutable, secured type etc are present or not). - warning (to be generated through schema json) custom type : warning
- Error wont be thrown if an unknown element is added.
- Check if template_name, template_version, template_type, template_provider are present in the source code (Validation framework has added this check) - error
- Cloud connection checked - warning if not found
- Rest Hook - url and headers - warning if not found
- If 'longDescription' in overview has length more than 400 chars. - warning
- When service 'description' has length more than 150 characters. - warning
- When feature 'bullet's description' has length more than 150 characters. - warning
- When title of the decision, that is, `decision.title` value is missing. - warning
- The `decision.mapped_parameter` value is missing from SCL - warning
- Parameters of email notification like title and receiver list key or value is missing. - warning - The Rest Hook URL is missing for create/on_delete. - warning
- The Resthook URL value is missing from scl for create/on delete - warning
- Templates title key and value are missing. - warning

Validations at the time of service publish:

- All the checks present in case of service creation.
- Check if atleast 1 executable section (resthook/templates/email notification) is present in the source code. - error
- Check the data type defined for each template variable and verify if values provided in the source code are compliant. - error
- Check if template name is valid or not - error
- Check if a valid cloud connection name is present. - error
- Check if dynamic binding for parameters is done correctly or not. - error - Check if mapped parameter in decision is present in the source code or not. -error - Check if Rest Hook section has "url" and "method" section. - error - Check if email notification has "recieverlist" and "body" elements. - error - Check if "subject" is mandatory in email notification. - error
- Validate template parameter type provided in the source code with the one defined in template : boolean, password type parameter should have string value, map type value should be of object type, counter type value should be a number of type counter, valid value as per regex defined, valid value as per options defined, catch missing parameters where value is required, check parameter mapping in template parameter. - error
- The Activity Ids in flow.sequence should be unique. - error
- The Activity Ids should be valid in `flow.sequence`. - error
- The Activity Ids in `flow.conditions.decisions.options.sequence` should be unique. - error
- The Activity Ids should be valid in `flow.conditions.decisions.options.sequence`. - error
- Pricing Parameter should be valid. It should be from plan parameter list - error
- Operator used in the pricing object should be valid. Use from gt/lt/ge/le/eq/members - error
- Condition validation for pricing object. - error
- Base condition in pricing object should be validated . Use any/all only. - error
- Plan should not have multiple pricing. - error
- Method and Headers are a must when URL is present in resthook. - error
- Rest Hook - create without URL or `on_destroy.url` should not be provided. - error
- Method and Headers are a must when `on_destroy` URL is present in resthook. - error
- Error Flow can not have Resthook with destroy enabled. - error
- Action array should have 1 type 'provision' . - error
- Validate mapped parameter in `input_parameters` - error
- Interpolation Validation - error
- Template does not have cloud connection name - error

Viewing service details

After you create a service, you can view its details from the Services Library page.

Procedure

1. In the navigation bar, click **Library > Services**.

   In the services page, you can see a list of all the services. Each service record has name, number of versions, and the date modified.

2. Search for your service by using any of the following search filters:

   - Enter your service name in **Search services**.
   - Click the show filter menu icon to filter records based on the status of a service. The different statuses of a service are **Draft**, **Published**, **Retired**.
   - You can also filter the services by clicking **Show globally accessible**. You can also include/exclude the globally accessible services by checking/unchecking the **Show globally accessible**.
   - You can click **Name** and **Date modified** column headers to sort records.
   - You can select a namespace from the **All namespaces** drop-down list. The list of services in the page changes based on the selected namespace. You can select "All namespaces" to see all of the services for which you have access.
   - You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.

3. You can select the following actions from the overflow menu of a service: **Add version**, **Duplicate**, **Rename service**, **Change Category** and **Delete**.

4. Expand the service to view the following details: **Version name**, **Status**, **Associated instances** and **Created**. You can select the following actions from the overflow menu:

   - Draft : Edit/Copy/Deploy/Publish/Delete
   - Published: View/Copy/Deploy/Retire
   - Retired: View/Copy/Delete

5. Click the service for which you want to see the details.

   The name of the service and the status of the service is displayed. From the overflow menu, you can **Add Version**, **Duplicate**, **Change category** and **Delete** actions. This menu changes as per the status of the service version.

- The **Overview** tab of the service is displayed by default, and has the following sections:
  - Service details.
  - **Service version** - You can select the version of this service from the drop-down list.
  - **Features** - You can order the service using the chosen features.
- The **Versions** tab of the service has the following sections:
  - **Default version** - You can change the default version of a service from this drop-down list.
  - **Search** version based on the status of a service version **Draft**, **Published**, **Retired**.
  - You can click **Version name** or **Created** column headers to sort the records.
  - Each version has the following actions from the overflow menu: **View details**, **Edit details**, **Edit version** and **Delete**.
  - You can click a service version to edit it. For service details, see Creating a Service.
- The **Associated instances** page displays the details of all the instances that are associated with a particular version of a service. In this tab, click the **Actions** overflow menu of the associated instance record to **View Details** of that instance, **Edit**, **Custom actions**, **Retry**, **Resume**, **Terminate** or **Delete**. The menu options are based upon the status of the associated instance.

Importing a service

You can import a service from your git repository.

About the task

You can import a service that is in service composition language version V3. However if you try to import a service that is in service composition language version V1 or V2, then an error message is displayed.

After importing a service, a service gets created with all the versions. If the service does not have any versions, it gets imported with a default version *1.0.0.0*.

Note: Services are provided by Managed services in the Manage service library of GIT repository for you to download and use. For the entire list of services, see Managing services in service library.

Before you begin

Before you import a service, do the following tasks:

- Fix the service definition that has a non-existent template version.
- Ensure that the service is free of any errors.

Procedure

1. In the navigation bar, click **Library** > **Services**. The Manage service library is displayed.
2. Click **Import service**.
3. Select either of the following options in **Assign access** section:
   - **Make this service globally accessible** - Services created globally are accessible to all users.
   - **Make this service part of a namespace** - Associate the service with a specific namespace. For this option, you must select a **Namespace** from the drop-down list to which the template must be assigned.

     Based on the value selected in the namespace drop-down list on the **Services** main page, the **Assign access** and **Namespace** values are automatically selected. Modify the values if you want.
4. From **Import service source** drop-down list, select any of the following values:
   - **GitHub**
   - **GitLab**
   - **Bitbucket server**
5. Enter the following details of your selected repository:
   - For both **GitLab** or **GitHub**, enter the following details:
     - Enter a repository URL, an access token, the relative directory of your service, and the related branch.

       The service must be in JSON format. For example, if the service is stored in a GitHub repository at **`vsphere/vsphere-example.json`**, then you must specify the following values:

- **GitHub repository URL**: `https://github.com/geekbass/terraform`
- **GitHub access token**: It is the token required to access GitHub template files. You can find it in your GitHub user account, and is not required for public projects. For example, `aaa`.
- **GitHub repository sub-directory**: `vsphere`
- **GitHub reference (branch or tag)**: `master`

- For **Bitbucket server**, enter the following details:

  - **Bitbucket repository URL** - The location where Bitbucket service files can be referenced from.
  - **Bitbucket access token** - The token used to access the Bitbucket service files.
  - **Bitbucket repository sub-directory** - The path relative to the Bitbucket location where service files can be referenced from.
  - **Bitbucket reference** - Select **Branch** or **Tag** option and enter the name of the tag or branch.

6. Click **Import**. The import is successful unless there are errors in the service.

Editing a service

You can edit a service from the Manage service library page.

About the task

You cannot edit published services. To modify a published service, make a copy of the service and then modify it.

Procedure

To edit a service, follow these steps:

1. In the navigation bar, click **Library** > **Services**. The Manage service library is displayed.
2. Search for your service by using any of the following search filters:

   - Enter your service name in **Search services**.
   - Click the show filter menu icon to list services based on their status.
   - You can click the column headings of **Name** and **Created** to sort the list.
   - You can also select the provider or type to list services based on your selection.
   - You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.
   - You can select a namespace from the **All namespaces** drop-down list. The list of service in the page changes based on the selected namespace. You can also select "All namespaces" to see all the service for which you have access.

3. Expand the service to see the list of versions. For the service version record you want to modify, select **Edit** option in the **Actions** overflow menu of a version.
4. Edit service details. For service details, see Creating a service. From the Edit service actions overflow menu, you can do the following actions:

   - Draft : Edit/Copy/Deploy/Publish/Delete
   - Published: View/Copy/Deploy/Retire
   - Retired: View/Copy/Delete

5. Save the changes.

Assigning access to a service

Procedure to assign access to multiple namespaces of a service.

Procedure

1. In the navigation bar, click **Library** > **Services**. The Manage service library is displayed.
2. In the overflow menu of the service, click **Assign access**.
3. In the **Assign access** dialog box, select the namespaces. You can select a single or multiple namespaces to which the user has access.
4. Click **Assign**.

Scenarios

- You can set or change a single or multiple valid namespaces to a service. The valid namespaces are those which the user has access to.

- A change in service namespace reflects in all its versions too.
- A service with multiple namespaces can be deployed only in any of the user accessible namespaces to which a service belongs.
  - **Example**: A service `S` belongs to namespaces `N1`, `N2`, and `N3`. User `U` has access to `N2`, `N3`, `N4`. Allowed namespaces for deployment are `N2` and `N3`.
- For a service, removal (multiple namespaces), change (single namespace) or add (Global to single or multiple namespaces) is allowed if the namespace of the associated service instance is present in the new namespace list.
  - **Example 1**: If service `S1` (globally accessible) is deployed in namespaces `N1` and `N2` (one service instance in `N1` and one in `N2`) then you can allow to update the namespaces of `S1` only if it includes `N1` and `N2`. That is, `S1` can be updated to `N1` and `N2` but not to `N2` and `N3`.
  - **Example 2**: If a service instance `SI` of service `S` is deployed in namespace `N2`, then service `S` won't be allowed to change namespaces from `N1`, `N2`, and `N3` to `N1` and `N3`, removing `N2` until service instance `SI` is deleted.
- Cloud connections, shared parameters, and templates should either be global in nature or of a namespace to which a service belongs (service instance, in the case of SMTP connection).

Note: It is recommended to avoid composing a service with multiple namespaces that has templates or cloud connections from multiple namespaces. Even though the service belongs to multiple namespaces, it can be deployed in only one namespace and the activities by using or belonging to other namespaces fail.

Duplicating a service

Procedure to duplicate a Managed services service and all of its versions.

About the task

There are two ways by which you can access the duplicate feature:

- Click **Duplicate** in the overflow menu of a service record in the **Library** > **Services** page.
- In the **Overview** page of service deployment, click **Duplicate** from the overflow menu.

Procedure

1. In the navigation bar, click **Library** > **Service**. The Manage service library is displayed.
2. In the overflow menu of the service you want to duplicate, click **Duplicate**.
3. Enter the **Service Name**.
4. In the **Duplicate service** page, select either of the following options in **Advanced > Assign access** section:
   - **Make the service globally accessible** - Services created globally are accessible by all users.
   - **Make this service part of a namespace** - Associate the service with a specific namespace. For this option, you must select a **Namespace** from the drop-down list to which the service must be assigned.

   Note: If you create a service in a specific namespace, then you can deploy the service only to that namespace.
5. Click **Duplicate**. All the versions of this service are duplicated as Draft.
6. Modify the details of the service based on your requirement. For more information to update the tabs, see Editing a service.
7. Click **Save**.

Changing a service category

You can change a service category from the Manage service library page.

To change a service category, follow these steps:

1. In the navigation bar, click **Library > Services**. The Manage service library opens.
2. Search for your service by using any of the following search filters:
   - Enter your service name in **Search services**.
   - Click the show filter menu icon on the right of the search instance filter to list services based on their status.
   - You can click the column headings of **Name** and **Created** to sort the list.
   - You can also select the provider or type from the list to filter services based on your selections.

- You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.
- You can select a namespace from the **All namespaces** drop-down list. The list of services in the page changes based on the selected namespace. Along with the records of the selected namespace, all globally accessible records are available in the list. You can select "All namespaces" to see all of the services for which you have access.

3. Select **Change category** option from the overflow menu for the service record you want to change category.
4. In the **Change category** window, select a category from the **Category** drop-down list. There must be more than one category to move the service. From this page, you cannot create a new category.
5. Click **Change category**. The category of the service is changed based on your selection.

Deleting a service

You can delete an existing service from the Manage service library.

Procedure

1. Go to the Manage service Library page.
2. From the overflow menu of a service record, select **Delete**.
3. In **Delete service** page, enter "delete" and click **Delete**.
4. All the versions associated with that service gets deleted.

   Notes:

   - The default version of the service can not be deleted.
   - Service deletion deletes all the versions of the service if there are no associated instances.

Saving a service to different repositories

You can create or edit a service and save it to a GitHub, GitLab, or Bitbucket Server repositories.

Procedure

1. From the overflow menu of the service, click **Push to GIT**.
2. In the **Push Service To Git** page, select any of the following values from the **Import template source** drop-down list:

   - **From GitLab**
   - **From GitHub**
   - **Bitbucket Server**

3. Enter details of the selected repository:

   - For **From GitLab** or **From GitHub**, enter a repository URL, an access token, the relative directory of your service, and the related branch. The service must be in JSON format. For example, if the service is stored in a GitHub repository at **`vsphere/vsphere-example.json`**, you must specify the following values:

     - **GitHub Repository URL**: `https://github.com/geekbass/terraform`
     - **GitHub Access Token**: `aaaaaaaaaa` It is the token required to access GitHub template files. You can find it in your GitHub user account.
     - **GitHub Repository path to service**: `vsphere`
     - **GitHub Ref (branch)**: `master`
     - **Commit Message** - Enter a commit message to push the changes into the GIT hub repository.

   - For **Bitbucket**, enter the following details:

     - **Bitbucket Repository URL** - The location where Bitbucket service files are stored.
     - **Bitbucket Access Token** - The token used to store the service in Bitbucket.
     - **Bitbucket Repository sub-directory** - The path relative to the Bitbucket location where the service is stored.
     - **Bitbucket Reference** - Select **Branch** or **Tag** option and enter the name of the tag or branch.

4. Click **Push to GIT**.

Service level actions

The following actions are handled at the service level.

Tasks that you can perform to manage services are as follows:

- Creating a service
- Viewing service details
- Assigning access to a service
- Editing a service
- Importing a service
- Duplicating a service
- Changing category
- Deleting a service

Service version level actions

The following actions are handled at the service version level.

Tasks that you can perform to manage service versions are as follows:

- Adding a service version
- Copying a service version
- Deleting a service version
- Publishing a service version
- Deleting a service version
- Retiring a service version

Adding a service version

You can create multiple versions of a single service, providing flexibility when managing a service and the supporting end user needs.

About the task

There are three ways by which you can add a service version:

- Click **Add version** in the overflow menu of a service record in the **Library** > **Services** pane.
- In the **Overview** pane of a service, select **Add version** from the overflow menu.
- **Version details** section on Overview tab when you create/edit service.

Procedure

1. In the navigation bar, click **Library** > **Services**. The Manage service library is displayed.
2. In the overflow menu of the service you want to add version, click **Add version**.
3. Enter the **Version name**.
4. Select the version from the drop-down list of **Based on** version.
5. Click **Add**.
6. Modify the details of the service based on your requirement. For more information to update the tabs, see Editing a service.
7. Click **Save**.

Copying a service

You can copy a service version to a new service.

About the task

You can copy a retired service version record to new service. The copied version is in Draft mode, you can further edit it using Edit service option.

Procedure

1. Go to the **Library** > **Services** pane.
2. Expand the service to see the service versions or click on the service to open the service details pane.
3. From the overflow menu of a retired service version record, select **Copy** option.
4. Enter the new service version in the **Name** field.
5. Enter the service name in the **Copy to sestination**.
6. From the advanced options, select the **Assign access**.
7. Click **Copy**.

Editing a service version

You can edit version details and also can upgrade/downgrade service instance to different service versions.

About the task

Notes:

- The upgrade/downgrade and edit flow is applicable only for active instances. If the active instance is in Error state, then Retry/Resume is performed first before upgrade/downgrade of a version.
- The edit flow of an active instance only considers the changes in input-parameter values, changes in the activity definition are not considered. If you need to apply changes in the activity definition, upgrade or downgrade flow must be used.
- All the defined activities are executed with the new input parameters values:
    - In case of unchanged activity, a modify/plan apply action is executed using new input parameters values.
    - In case you removed an existing activity and specified a new one, the previous activity is destroyed and the new one installed on upgrade or downgrade only.

Procedure

1. In the navigation bar, click **Library** > **Service**. The Manage service library is displayed.
2. In the overflow menu of the service you want to edit, click **Edit**.
3. In the **Version details** section on Overview tab, click **Edit details**.
4. Modify the **Version name**.
5. In the **Upgrade** tab, select the version to be upgraded. You can upgrade from a lesser than a current version.
6. In the **Downgrade** tab, select the version to be downgraded. You can downgrade from a higher version.
7. Click **Save**.

Publishing a service version

You can publish a service version and perform operations on it.

Procedure

1. Go to the **Library** > **Services** pane.
2. Expand the service record to see all the versions.
3. From the overflow menu of a service version record, select **Publish** option. If there are errors in your service version, then **Service failed to publish** message is displayed with details of the error. Fix those errors and retry the publish operation. Alternatively, you can open the service in edit mode and click **Publish**. If your service has validation errors, then the **Publish** button is disabled.
4. In **Publish service** confirmation message, click **Publish** to proceed. On successful completion of publish, a confirmation message is displayed. The published service is displayed in the Service library.

   Note: In **Library** > **Services** pane, a published service version has **View**, **Copy**, **Deploy**, and **Retire** options in the actions overflow menu. You cannot delete or edit a published service version, but you can retire the service version and delete it. After you retire a service version from the Manage service library, refresh the library page for the status to show up in the user interface.
5. To view and deploy the published service, click **View published service** in the success confirmation message.

In the **Library** > **Services** pane, click **View** option or **Deploy** option in the overflow menu of the published service record to view or deploy the service respectively. You can also deploy the published service from the **View** page. To create a service instance, see Creating a service instance.

Deleting a service version

You can delete an existing service version from the Manage service library.

Procedure

1. Go to the Manage service Library page.
2. Expand the service record to see the list of versions.
3. From the overflow menu of a service version record, select **Delete**.
4. In **Delete service version** page, enter "delete" and click **Delete**.

   Notes:

   - A default service version cannot be deleted. To delete a default version, make other version as default version.
   - The deployed instances that are associated with this version cannot be modified after deleting service version.
   - In a service where upgrade/downgrade path is mentioned for a particular version, on deleting that version, an error message is displayed to the user. The user needs to manually remove the entries from the source code for the missing version.

Retiring a service version

You can retire a published service version.

About the task

You cannot retire a draft service version from the user interface, however, you can retire it by using API commands. For the API details, see Service and service instance APIs.

You can retire a particular service version by using the following methods:

- Manually retire a service version
- Auto-retire a service version

Manually retire a service version

1. Go to the **Library** > **Services** pane.
2. Expand the service to see the service versions or click the service to open the service details pane.
3. From the overflow menu of a published service version record, select **Retire** option. On successful completion, a **Retire service** confirmation message is displayed.
4. In the **Retire service** confirmation message box, click **Retire**. The status of the record changes to "Retired".

   Note: The overflow menu of a retired service version has **View**, **Copy**, **Delete** options.

Auto-retire a service version

You can set a retirement date for a published service version. The service version is retired on that particular date and the status of service version is changed to `Retired`. This is useful where you want to keep a service version available for the user to deploy for a certain period only. A retired service version cannot be deployed.

1. Go to the **Library** > **Services** pane.
2. Expand the service to see the service versions or click the service to open the service details pane.
3. From the overflow menu of a published service version record, select **Set retirement** option.

4. Select the retirement date.
5. Click **Save**.

Unretiring a service version

You can unretire a retired service version.

About the task

For the API details, see Service and service instance APIs.

Procedure

1. Go to the **Library** > **Services** page.
2. Expand the service to see the service versions or click on the service to open the service details page.
3. From the overflow menu of a retired service version record, select **Unretire** option. On successful completion, a **Unretire service version** confirmation message is displayed.
4. In the **Unretire service version** confirmation message box, click **Unretire**. The status of the record changes to "Published".

Managing service instances

Create and manage the service instance details. Terminate or delete the service instances that are not needed anymore.

- Creating a service instance
- Creating bind instances for service instances
- Managed services Kubernetes resource controller for services
- Editing a service instance
- Upgrading or downgrading a service instance
- Viewing service instance details
- Terminating and deleting a service instance
- Resuming or retrying a service instance

Creating a service instance

You can search and create service instances of published offerings from the Managed services catalog. You can customize the service instances based on your requirement and use it in the overall application development.

Before you begin

If you are using tags in your service, see Prerequisites for tags.

Procedure

To create a service instance, follow these steps:

1. In the navigation bar, click **Library > Services**. The Manage service library is displayed.
2. Search for your service by using any of the following search filters:
   - Enter your service name in **Search services**.
   - Click the show filter menu icon to list services based on their status.
   - You can click the column headings of **Name** and **Created** to sort the list.
   - You can also select the provider or type to list services based on your selection.
   - You can use the pagination drop-down to select the number of records you want to display in a page at a time.

- You can select a namespace from the **All namespaces** drop-down list. The list of services in the page changes based on the selected namespace. You can select "All namespaces" to see all of the services for which you have access.

3. Click **Deploy** in the overflow menu of your service tile.

   As an alternative, you can click the service to preview its **Overview** and **Associated instances** details.

   The **Associated instances** page displays the details of all the instances that are associated with the service. In this tab, click the **Actions** overflow menu of the associated instance record to **View details** of that instance.

4. Go through the details of the service. You can see the following details of the service in the **Overview** page:

   - Chart version, published date, and type
   - **Details** section has a long description of the service.
   - For additional information about the service, see the **Useful links** section. It has **Documentation** and **Support link**.

5. Select your plan and click **Next**.

6. Enter the **Name** of the service instance.

   Notes:

   - The service instance name must begin with an alphabet and must have a combination of only these characters: [a-zA-Z][a-zA-Z0-9_-].
   - If service instance has an application activity then the service instance name should consist of lower case alphanumeric characters, '-' or '.', and must start and end with an alphanumeric character.

7. For globally accessible services, you can select a namespace from the **Namespace** drop-down list. For other namespaces, the option is not enabled for selection.

8. Consider the following for dynamic cloud connections in the service definition:

   - If you have defined a parameter of type connection in the service definition and linked it to the **Cloud connections** parameter value, then the parameter is displayed here as a drop-down list. For example, you have created "Conn" as a parameter of type connection, then "Conn" drop down list is displayed. For the complete input_parameter format for "conn" definition in the **Source code** tab, see Different ways of adding values for the cloud connection in Templates topic. This drop down list fetches all global cloud connections of the provider mentioned in the input_parameter. For example, if the provider is VMware vSphere, then the drop down list includes all global cloud connections of VMware vSphere provider. If you change the **Namespace** value, the drop-down value contains a list of global cloud connections that are tagged to the selected namespace and are associated to the provider.

9. Consider the following for dynamic shared parameters in the service definition:

   - If you have defined a parameter of type sharedparameter in the service definition and linked it to a Data Object value, then the parameter is displayed here as a drop-down list. For example, you have created "shared" as a parameter of type sharedparameter, then "shared" drop down list is displayed. For the complete input_parameter format for "shared" definition in the **Source code** tab, see Shared parameters in Templates topic. This drop down list fetches all globally defined Data Objects of the Data Type that is mentioned in the input_parameter. If there are more than one Data Object for the selected Data Type, then you must define a default Data Object. By default, this default value is preselected for the "shared" parameter drop down. If you change the **Namespace** value, the drop-down value contains a list of global Data Objects that are tagged to the selected namespace and are associated to the Data Type.

10. Enter the values for the selected plan parameters.

    Note: If a value is not available for a non mandatory service or activity parameter referenced within a plan parameter, the service instance fails with an error message. For example, you have referenced the `cc` parameter of an email notification activity in a plan parameter. If the value for the non-mandatory `cc` parameter is not available, then the service instance fails with the following error message:

    ```
    Service deploymentError: Invalid resource : ${parameters.cc} (hint: 'cc' not found)
    {"error":{"statusCode":500,"message":"Internal Server Error"}}
    ```

11. If you have any tags created for the service, then select your tag.

12. Click **Create instance**. A message confirming the successful submission of a service instance request is displayed. You can go to **Deployed instances** page to view the status of the current deployment or go to **Library** to create more instances.

Results

- If your service instance completes successfully, then the deployed instance status changes from `In Progress` to `Active`. If an activity component of a service instance fails, then the deployed service instance is marked as error.
- If you use a long running Business Process Manager template in your service that requires human intervention for claim and approval, the status of the service instance deployment changes to "Wait" state and in the activity instances, the status changes to "Waiting" state. To know the state and piid number, go to **Log file** tab > **Activity log** of the service instance. This piid is the stack id in Managed services. The Business Process Manager administrator logs in to the ProcessPortal dashboard to claim and approve the pending tasks. When the polling activity detects that the process is complete, the service instance state moves from "Wait" to "In progress" and then subsequently to the "Active" state based on the service flow. The activity instances of the service instance also move from "Waiting" to "Created" state.

- If SMTP connection is used in your service, then a primary connection for the selected namespace is retrieved first and in the absence of that connection, search is done for a globally accessible primary connection. If both primary connections do not exist, then a secondary connection is attempted in the same sequence as the primary connection. In the absence of both primary and secondary connections, an error message is added to the logs.

Creating bind and custom action instances for service instances

From the deployed service instance page, you can perform the bind or custom action.

About this task

Limitations:

- You can create any number of bind instance records for a service instance. However, only one bind instance can be created at a time. - You must wait for a bind instance to complete before you create another.
- Bind is available only for services instances that are in active state.
- You cannot deploy a bind instance before the service instance creation is complete.

For more bind related limitations, see Troubleshooting Service Composer or Portal issues.

Procedure

1. Navigate to the deployed instances page, by using one of these methods:
   - In the navigation bar, click **Deployed instances** > **Services** tab.
   - Alternatively, any operational user can access the instances page through the service library. From the IBM Cloud Pak for AIOps console, click **Automate infrastructure** > **Service library**. The **Instances** tab shows the deployed instances.
2. Search for your service instance.

   Note: You can also use the following search filters:

   - Enter the name of the instance in the search instance filter.
   - Click the show filter menu icon to filter records based on the status of a record. The different statuses of a service instance are **Active**, **In progress**, **Error**, and **Terminated**.

     Note: The **Wait** status is available whenever you use long running Business Process Manager template in your service.
   - You can click **Name** and **Ordered on** column headers to sort records. The **Name** also includes the namespace details. In addition, you have **Service offering** and **Status** column headers.
   - You can select a namespace from the **All namespaces** drop-down list. The list of service instances in the page changes based on the selected namespace. You can select "All namespaces" to see all of the service instances for which you have access.

   You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.
3. There are two ways to select **Custom actions**:
   - From the overflow menu of the service instance, select **Custom actions**.
   - Click on the service instance to view the details and select **Custom actions** from the overflow menu.
4. A **Custom actions** dialog box opens with the list of bind and custom actions.
5. Select the action and click **Next**, a service deployment page is displayed. It is called **Bind service instance** page for bind action.
6. Optionally, update the parameter values as required.
7. For Bind action, Click **Bind instance**. For custom actions, click action name button for executing this custom action. A new instance record is included in the **Overview** tab > **Custom action instances** section of the deployed instance. In the **Overview** > **Action** section, you can view the status of the operation. Click **View log** to view the log information of the bind action.
8. You can select **UnBind** action from the overflow menu of the bind instance records. The bind instance record changes from Active state to Terminated state.
9. From the **Overflow** tab > **Action** section, you can track the unbind operation. Click **View log** to view the log information of the unbind action.

Managed services Kubernetes resource controller for services

The Managed services Kubernetes custom resource definition and controller provides you with the ability to deploy and delete Managed services service instances using Kubernetes yaml resource files. With this capability, a single Kubernetes yaml can be created to deploy a composite application that consists of both Kubernetes deployable resources and traditional infrastructure resources, which are deployed with Terraform in Managed services. A typical containerized application is an example of a Kubernetes deployable resource. Examples of traditional infrastructure resources are virtual machines, storage, and so on.

About the task

The Managed services Kubernetes resource controller deploys the Managed services service and waits for Active status. If the deployment FAILS, then the controller continues to issue Resume requests until the service becomes Active.

Before you begin

A role binding must be defined for a user or group to have the authority to deploy a service in Managed services.

The following is an example for the **icp:camgroup:operator** user group:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
   name: icp:camgroup:operator:service
roleRef:
   apiGroup: rbac.authorization.k8s.io
   kind: ClusterRole
   name: cam-controller-manager-role
subjects:
- apiGroup: rbac.authorization.k8s.io
   kind: Group
   name: icp:camgroup:operator
```

Procedure

- Deploying a Managed services service
- Displaying the status of a Managed services service
- Deleting a deployed Managed services service

Deploying a Managed services service

1. Go to your Kubernetes environment that is same as Managed services.
2. Create your YAML file. The file allows the definition of the service deployment to be placed under the source control that allows changes to be tracked. An example Managed services Kubernetes service resource controller yaml file:

```
apiVersion: cloudautomationmgr.ibm.com/v1alpha1
kind: Service
metadata:
  name: service-test-vsphere-1
  namespace: "default"
spec:
  connection: "camDc2Rtp"
  service:
    name: "LrhNodeOnSingleVM-vsphere"       //Instance that you are going to deploy
    instance_plan: "host-164"
    action: "Provision"                     //The action is provision.
    parameters:
      templateParameters: []
      specifications:
        parameters:                         //The parameters that are passed with the instance
          vSphereConfig: vSphere-CAMDC2
          vm_ipv4_address: "9.9.9.9"
          vm_name: "camc-vis232c-vm-164"
          vm_dns_servers:
          - "9.9.9.9"
          vm_dns_suffixes: ["rtp.raleigh.ibm.com"]
```

   - parameters - It is a collection of name-value pairs that map to the Managed services Service Input Parameter Key and Value. For example:

```
parameters:
  vSphereConfig: vSphere-CAMDC2
  vm_ipv4_address: "9.9.9.9"
  vm_dns_suffixes: ["rtp.raleigh.ibm.com"]
```

- **secretParameters** - (Optional) List of Managed services secret parameter data objects to use with the deployed service. For example:

```
secretParameters:
        name: vm_password_lrh
        secretKeyRef:
                name: vm-password-lrh-secret
                key: password
```

where

name: name of service parameter

secretKeyRef:

name: Name of the Kubernetes secret

key: key is the name of the secret data defined in the secret

3. Run the following kubectl command to deploy a Managed services service:

```
kubectl apply -f <yaml file name>
```

Displaying the status of a Managed services service

1. Verify the status of a deployed Managed services service by using either of the following commands:

```
kubectl describe -f <yaml file name>
```

Or

```
kubectl describe services.cloudautomationmgr.ibm.com service-test-vsphere-1 -n default
```

Example:

```
 Name:          service-test-vsphere-1
 Namespace:     default
 Labels:        <none>
 Annotations:   kubectl.kubernetes.io/last-applied-configuration:
                  {"apiVersion":"cloudautomationmgr.ibm.com/v1alpha1","kind":"Service","metadata":
{"annotations":{},"name":"service-test-vsphere-1","namespa...
 API Version:   cloudautomationmgr.ibm.com/v1alpha1
 Kind:          Service
 Metadata:
   Creation Timestamp:  2019-08-09T19:50:52Z
   Finalizers:
     Service.cloudautomationmgr.ibm.com
   Generation:         2
   Resource Version:   6549804
   Self Link:          /apis/cloudautomationmgr.ibm.com/v1alpha1/namespaces/default/services/service-
test-vsphere-1
   UID:                feadd3c9-bade-11e9-a019-00163e01cbab
 Spec:
   Secrets:
     Env:
   Service:
     Action:          Provision
     instance_plan:   Standard
     Name:            LrhNodeOnSingleVM-vsphere
     Parameters:
       Specifications:
         Parameters:
           V Sphere Config:  vSphere-CAMDC2
           vm_datacenter:    CAMDC2
           vm_dns_servers:
             9.42.106.3
           vm_dns_suffixes:
             rtp.raleigh.ibm.com
           vm_ipv4_address:  9.46.126.166
           vm_name:          camc-vis232c-vm-166
 Status:
   Action:  provision
   Conditions:
     Last Transition Time:  2019-08-09T19:53:43Z
     Reason:                Active
```

```
    Status:                 True
    Type:                   provision
  Message:
  Service ID:               5d4dce9d5b972d001dc804d8
  Status:                   Active
Events:
  Type    Reason                  Age    From                     Message
  ----    ------                  ----   ----                     -------
  Normal  provision:Submitted     2m51s  cam-controller-service   provision status changed for
default/service-test-vsphere-1
  Normal  provision:In Progress   2m32s  cam-controller-service   provision status changed for
default/service-test-vsphere-1
  Normal  provision:Active        1s     cam-controller-service   provision status changed for
default/service-test-vsphere-1
```

- The Status fields returned from the **`kubectl describe`** command:
  - Action - Last action performed, that is provision, resume, delete
  - Message - Last error message, blank if no error message.
  - Service ID - Managed services service ID of the deployment.
  - Status - Status of the last action performed.
  - Conditions
    - Last Transition Time - Last update time of the condition.
    - Reason - The status of the Managed services action.
    - Status - It is 'True' when the condition is valid.
    - Type - The type of the condition - the Managed services action (For example, provision, resume, delete).
  - Outputs
    - Name - Name of the output parameter.
    - Type - The type of the parameter, that is, string, list, map.
    - Value - The value of the output parameter in JSON format.
- The Event fields returned from the **`kubectl describe`** command are Type, Reason, Age, and From. Example:

```
Type    Reason                  Age    From                     Message
----    ------                  ----   ----                     -------
Normal  provision:Submitted     6m53s  cam-controller-service   provision status changed for
default/service-test-vsphere-1
Normal  provision:In Progress   6m34s  cam-controller-service   provision status changed for
default/service-test-vsphere-1
Normal  provision:Active        4m3s   cam-controller-service   provision status changed for
default/service-test-vsphere-1
```

2. Optionally, you can also use Managed services Kubernetes controller startup parameters - controller parameter. They are defined in **`cam-controller-statefulset.yaml`** file. The args parameters are specified in the pod definition.

   The **`-syncPeriod`** is a controller parameter for the time duration between resource status checks. Value units are 's', 'm', 'h'." (Default is 2m).

   For example:

   - -syncPeriod=2h30m
   - -syncPeriod=300s
   - -syncPeriod=30m

Deleting a deployed Managed services service

If you no longer require the deployed instance of a service instance, run either of the following commands to delete it:

```
kubectl delete -f <yaml file name>
```

or

```
kubectl delete services.cloudautomationmgr.ibm.com service-test-vsphere-1 -n default
```

Note: The command deletes the Managed services instance. After the Managed services instance is successfully deleted, the Kubernetes resource that represents the Managed services instance is also deleted.

Editing a service instance

You can edit any active deployed service instance after selecting the required service instance and editing the parameters.

The following table shows how the different activities are managed during the Edit action when the Editable flag is checked for an activity:

Activity	Action on Edit
Terraform template	Modified (Depends on the template provider)
Rest Hook	Destroyed and re-created
Email notification	Executed again
Helm Chart	Destroyed and re-created
Decision	Not supported
BAW/BPM template	Not supported
Ansible credential	Destroyed and re-created
Ansible inventory	Destroyed and re-created
Ansible template	Executed again

Notes:

- The activity parameters, which are parameterized by using the service input parameters can only be edited by using this feature.
- If you change an activity instance from `Editable` to `non-Editable` or vice versa after the service is deployed, then Edit service instance action is not affected due to this change.
- If there are no input parameters or if the values of input parameters are not changed during the edit service instance, the edit action does not take place.
- If a service includes Helm and/or Ansible job activities, after editing the deployed service instance where input parameters are changed for other activity and not for Helm and Ansible job activities, the Helm and Ansible activities are also destroyed and re-created.
- In case where one activity depends on another activity, after an Edit action, if the first activity is destroyed and re-created then the other activity also gets destroyed and re-created. If you want to edit only the first activity, make the other activity as non-editable so that it does not participate in the Edit action. After Edit action, only the first activity gets destroyed and re-created.

To edit a service instance:

1. Navigate to the deployed instances page, by using one of these methods:
   - In the navigation bar, click **Deployed instances** > **Services** tab.
   - Alternatively, any operational user can access the instances page using the service library. From the IBM Cloud Pak for AIOps console, click **Automate infrastructure** > **Service library**. The **Instances** tab shows the deployed instances.
2. In the Deployed Instances page, you can see a list of all deployed instances. Each instance record has name, service offering, status, and the time of order.
3. Edit the deployed active instance by using any of the following methods:

   a. Search for your deployed instance by using any of the following search filters:

   - Enter your instance name in **Search instances**.
   - Click the show filter menu icon to filter records based on the status of a record. The different statuses of a service instance are **Active**, **In progress**, **Error**, and **Terminated**.

     Note: The **Wait** status is available whenever you use long running Business Process Manager template in your service.
   - You can click **Name** and **Ordered On** column headers to sort records. The **Name** also includes the namespace details. In addition, you have **Service Offering** and **Status** column headers.
   - You can select a namespace from the **All namespaces** drop-down list. The list of deployed services in the page changes based on the selected namespace. You can select "All namespaces" to see all of the deployed services for which you have access.
   - You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.
   - Click the **Edit** in the overflow menu. You can edit parameters of any Active service instance.

   b. The service instance edit action can also be selected from the service instance details page.

   - Click the service instance to be edited.
   - Click **Edit** from the Actions overflow menu of the service instance details page.

   c. Alternatively, you can also edit the service instance by selecting the service first and then its associated instance.

   - From the menu bar, click **Library** > **Services**.
   - Click the wanted service from the list of services.
   - Click the **Associated instances** tab and select the instance and click **Edit** from the overflow menu.
4. Confirm **Edit** in the **Edit service instance** dialog box.
5. You can Modify or Delete the service tags. For more information about adding tags, see Creating a service.

   Notes:

   - You can modify only user added tags and not system tags.

- To **Delete** the service tag, click the delete icon on the right side of the text box.

6. Modify the required parameters. For more information about creating user-defined parameters, see Creating new input and output parameters.

   Notes: The service input parameters, which are marked as Editable during creation can only be updated here.

   If the **Editable** is not checked for an activity and the user edits the service instance, then the service input parameter values are updated in the service instance but in actual resource the parametrized property value is not be changed.

7. Click **Perform action**. The **Planned changes** page is displayed.

8. Verify the log output and then click **Edit instance**. The service instance is redeployed with the latest modifications.

Upgrading or downgrading a service instance

You can upgrade or downgrade a particular active service instance. The upgrade or downgrade service version must be selected while creating or editing a service version. See Adding a service version or Editing a service version for more details.

Note: You can upgrade or downgrade a particular active service instance only by using the plan selected during initial service deployment. Upgrade and downgrade of an active service instance from Plan A to Plan B is not supported.

The following table shows how the different activities are managed during the Upgrade/downgrade action when any value of the activity changes between versions:

Activity	Action on Upgrade/downgrade
Terraform template	Modified (Depends on the template provider)
Rest Hook	Executed again
Email notification	Executed again
Helm Chart	Destroyed and recreated
Decision	Executed again
BAW/BPM template	Executed again
Ansible credential	Destroyed and recreated
Ansible inventory	Destroyed and recreated
Ansible template	Executed again

Note: If the activity ID of an activity changes, the activity is destroyed and recreated.

Procedure

1. Navigate to the deployed instances page, by using one of these methods:

   - In the navigation bar, click **Deployed instances** > **Services** tab.
   - Alternatively, any operational user can access the instances page through the service library. From the IBM Cloud Pak for AIOps console, click **Automate infrastructure** > **Service library**. The **Instances** tab shows the deployed instances.

     In the deployed instances page, you can see a list of all deployed instances. Each instance record has name, status, service offering, user ordered, and the time of order.

2. Search for your deployed instance by using any of the following search filters:

   - Enter your instance name in **Search instances**.
   - Click the show filter menu icon to filter records based on the status of a record. The different statuses of a service instance are **Active**, **In progress**, **Error**, and **Terminated**.

     Note: The **Wait** status is available whenever you use long running Business Process Manager template in your service.

   - You can click **Name** and **Ordered On** column headers to sort records. The **Name** also includes the namespace details. In addition, you have **Service offering** and **Status** column headers.
   - You can select a namespace from the **All namespaces** drop-down list. The list of deployed services in the page changes based on the selected namespace. You can select "All namespaces" to see all of the deployed services for which you have access.
   - You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.

3. Click **Upgrade** or **Downgrade** from the overflow menu of the service instance. The other way to upgrade or downgrade is to click the service instance and click **Upgrade** or **Downgrade**. These options are available only when Upgrade or Downgrade is specified while adding a new service version or editing a service version.
4. Click **Upgrade** or **Downgrade** in the confirmation dialog box.
5. A **Upgrade service instance** page is displayed where you can edit service input parameters before upgrading or downgrading an instance.
6. A **Planned changes** page is displayed with the log output. Verify the output and click **Upgrade** or **Downgrade** if the log output looks ok.
7. The service instance is upgraded or downgraded to the desired service version.

Viewing service instance details

After you order a service, you can view its details from the Deployed Instance page.

Procedure

1. Navigate to the deployed instances page, by using one of these methods:
   - In the navigation bar, click **Deployed instances** > **Services** tab.
   - Alternatively, any operational user can access the instances page through the service library. From the IBM Cloud Pak for AIOps console, click **Automate infrastructure** > **Service library**. The **Instances** tab shows the deployed instances.

     In the deployed instances page, you can see a list of all deployed instances. Each instance record has name, status, service offering, user ordered, and the time of order.
2. Search for your deployed instance by using any of the following search filters:
   - Enter your instance name in **Search instances**.
   - Click the show filter menu icon to filter records based on the status of a record. The different statuses of a service instance are **Active**, **In progress**, **Error**, and **Terminated**.

     Note: The **Wait** status is available whenever you use long running Business Process Manager template in your service.
   - You can click **Name** and **Ordered on** column headers to sort records. The **Name** also includes the namespace details. In addition, you have **Service Offering** and **Status** column headers.
   - You can select a namespace from the **All namespaces** drop-down list. The list of deployed services in the page changes based on the selected namespace. You can also select "All namespaces" to see all deployed services for which you have access.
   - You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.
3. Click the service instance for which you want to see the details. Alternatively, you can click the **View details** in the overflow menu.

   The name of the order and status of the order is displayed. From the overflow menu, you can **Rename** an order name or **Reorder** the service.
   - The **Overview** tab of the instance is displayed by default, and it has the following sections:
     - Service offering section from where you can **Redeploy instance**.
     - Instance Details section that has information about the owner, order time, created time, and plan name.
     - Action section that has the statuses of all deployments and bind/custom operations. Click **View log** to view the log of an operation.
     - Activity Instances section that has a list of all template instances that are used by the service. When you create a service, you provide an instance name for the template in the **Composition** tab. The name of the instance is prefixed by the order name followed by the instance position. For example, if the order name is `R001` and it is the first instance of the template, then the name of the record in this section is `R001_00_<template instance name>`.
     - Custom Action Instances section that has all the created bind as well as custom instances along with their status. You can unbind a **Bind** action, from the overflow menu. For more information, see Creating bind instances for service instances. Every action has input or output parameters.
     - Output parameters tab lists all the output parameters of a deployed service.
     - Service parameters tab lists all the deployment parameters of the service.
   - Tags tab displays all tags that were created while defining the service.

- In the **Log file** tab, you can view the service deployment log file and the template deployment log file. Use the **Copy logs** if you want to copy the logs in to a text file.

Note: If the Log exceeds the maximum size of 200 KB, then only the last 200 KB of the Log is displayed.

Terminating and deleting a service instance

You can terminate a service instance from the deployed service instances. After you terminate a service instance, it goes to the `Destroyed` state and then the **Delete** action is enabled. If you delete the instance that is in `Destroyed` state, then all the IDs are cleaned up from Terraform jobs database and IaaS database.

Procedure

1. Navigate to the deployed instances page, by using one of these methods:
   - In the navigation bar, click **Deployed instances** > **Services** tab.
   - Alternatively, any operational user can access the instances page through the service library. From the IBM Cloud Pak for AIOps console, click **Automate infrastructure** > **Service library**. The **Instances** tab shows the deployed instances.
2. In the service instances list, click the **Actions** overflow menu of an active or error instance that you want to terminate. The action list is displayed.
3. Click **Terminate**.
4. In the Terminate Service confirmation window, click **Terminate**. All resources are destroyed from the Terraform and the service instance is in 'Destroyed' state.

   Success:
   - If all the activities / template of the deployed service instance are destroyed successfully, then the **STATUS** of the service instance changes from `Active` to `Terminated`. The **STATUS** of the templates in the template instance section changes from `Created` to `Terminated`.

   Failure:
   - If a template in the template instance section is already in `Failed` state, then the **STATUS** of the service instance remains in error state. The template remains in `Failed` state even after the completion of terminate operation.
   - Templates that are not destroyed during a service instance termination are never reattempted.

   For information about deleting template instances, see Deleting a template instance.
5. From the **Actions** overflow menu of a terminated service instance, click **Delete**. You can do a force delete on service instances that are in active or error states.
6. In the Delete Service confirmation window, click **Delete** for terminated service instances. For error or active service instances, enter "delete" in the confirmation window and then click **Delete** to force delete. A message confirming the initiation of delete action is displayed.
7. Click **Close** in the confirmation window. The successfully deleted service instance is no longer available in the deployed service instance list.

Note: If you attempt the `Terminate` action on the service instance, then the retry, resume, and bind options are not available in the user interface irrespective of the status of the terminate action.

Resuming or retrying a service instance

You can resume or retry a service instance that is in error state.

About this task

Resume and Retry actions are the redeployments of a service instance from the point of failure. However, Retry action destroys partially completed resources before proceeding with the deployment.

Note: If Managed services orchestration has multiple templates and a template fails in the workflow, then all subsequent templates are also not considered for deployment.

Example scenarios for a service that is defined to create two virtual machines:

- The deployment fails after it creates the first virtual machine. The Resume and Retry actions do not re-create the first virtual machine instead proceeds only with the second virtual machine.
- The deployment fails after it creates the first virtual machine and partially creates the second virtual machine. The Resume action resumes from the exact point of failure but Retry action destroys the partially created virtual machine and then proceeds with the redeployment.

Resume

Resume action works only with Terraform template. Only for Terraform template, it does plan and apply. For other activities, it performs retry action.

1. Navigate to the deployed instances page, by using one of these methods:
   - In the navigation bar, click **Deployed instances** > **Services** tab.
   - Alternatively, any operational user can access the instances page through the service library. From the IBM Cloud Pak for AIOps console, click **Automate infrastructure** > **Service library**. The **Instances** tab shows the deployed instances.
2. In the service instances list, click the **Actions** overflow menu of an error instance that you want to resume. The action list is displayed.
3. Click **Resume**. A **Resume service instance** confirmation box is displayed.
4. Click **Resume**. A **Resume service instance** page is displayed.
5. Select **True** to resume all the activities with updated values including those that are executed successfully. Select **False** to resume with values provided during the original deployment for the successfully executed activities and use the updated values from the point of failure.
6. Click **Resume instance**.
7. From the list of deployed instances, click **View details** for this record from the overflow menu. In the **Overview** page of the service instance, the status changes to "In Progress", and after successful resume operation, the status changes to "Active" state. If the deployment of the service instance fails, then an error message informing about the failure is displayed.

   Note: If you get a 401 error during resume operation, then clean the cache and retry.

Retry

1. In the navigation bar, click **Deployed instances > Services**.
2. In the service instance list, click the **Actions** overflow menu of an error instance that you want to retry. The action list is displayed.
3. Click **Retry**. A **Retry service instance** confirmation box is displayed.
4. Click **Retry**. A **Retry service instance** page is displayed.
5. Select **True** to retry all the activities with updated values. Select **False** to retry with values provided during original deployment for the successfully executed activities and use the updated values from the point of failed activity.
6. Click **Retry instance**.
7. From the list of deployed instances, click **View details** for this record from the overflow menu. In the **Overview** page of the service instance, the status changes to "In Progress", and after successful retry operation, the status changes to "Active" state. If the deployment of the service instance fails, then an error message informing about the failure is displayed.

Creating and managing custom action

Custom action helps to perform day-to-day actions on the provisioned service instance when attached to a service version.

For troubleshooting and limitations in the service composer, see Troubleshooting Service Composer or Portal issues.

Tasks that you can perform to custom actions is as follows:

- Creating a custom action
- Creating a custom action category
- Viewing custom action details
- Editing a custom action
- Assigning custom action to a service version
- Assigning access to a custom action
- Deleting a custom action

Creating a custom action

You can create a custom action from the Managed services service library page.

About the task

If there are any problems while you create, save, or lock your custom action, then warning or error messages are displayed.

Procedure

1. In the navigation bar, click **Library** > **Custom Action**. The Manage service library is displayed.
2. Click **Create custom action**.
3. In the **Create custom action** window, enter the following details:
   - Select either of the following options in **Assign access** section:
     - **Make this custom action globally accessible** - Custom actions created globally are accessible to all users.
     - **Make this custom action part of a namespace** - Associate the custom action with a specific namespace. For this option, you must select a namespace from the dropdown list to which the template must be assigned.

     Notes:
     - Based on the value selected in the namespace dropdown list of the **Library** > **Custom Action** window, the **Assign access** and **namespace** values are automatically selected. Modify the values if you want.
     - If the custom action is defined as globally accessible, then only globally accessible resources are available in the composer. For example, globally accessible cloud connections, shared parameters, and content runtimes are only available in the custom action composition tab for a custom action that is defined as globally accessible.
     - If the custom action is defined in a namespace context, then only globally accessible resources and resources defined in the same namespace context are available in the composition tab of the custom action.
   - Enter the custom action name.
   - Optionally, select a category.

     If you do not specify any category, then the custom action is assigned to Cloud Actions category with custom action. You can select **Add a category** from the **Category** dropdown list to create a new category. If you select **Add a category**, then enter the name of the new category and click **Create**. The **Select from list** allows you to go back to the original list of categories.

     For information about categories, see Creating a custom action category.

     Notes:
     - The composer does not allow you to create duplicate custom action names.
     - Custom action name must follow the Kubernetes naming convention, or else the locked custom action do not get listed in IBM Cloud Pak® for AIOps catalog:
       - Alphanumeric, underscore, hyphen, and space
       - Starts with an alphabet
     - If you plan to lock a custom action, ensure that its name is less than 64 characters long. If you have more than 64 characters, then locking the custom action to IBM Cloud Pak® for AIOps may cause issues.
4. Save your changes. The custom action is displayed in a `Unlock` status in the Manage service library. The overflow menu of the unlock service record in the Manage service library page has **Edit**, **Assign access**, **Lock**, and **Delete**. The **Save** button is disabled if there are validation errors in the service.
5. If you want to assign custom action to service version, see Assigning custom action to a service version

Creating custom action categories

Create a category to organize your custom actions in the library.

Procedure

To create a category, follow these steps:

1. In the navigation bar, click **Automate Infrastructure** > **Library**. Click the **Custom Action** tab.
2. In the **Custom Action categories** pane, click **Add category**.
3. Enter a name for the category and click **Add**. When the category is created, a confirmation message is displayed.

Alternatively, when you create a service, select the **Add a category** item in the **Category** drop-down to create a new category. The naming convention for category name is as follows:

- Must start with a letter

- May contain alphanumeric, underscore, hyphen, and space characters

Viewing custom action details

After you create a custom action, you can view its details from the Services Library page.

Procedure

1. In the navigation bar, click **Library > Custom Action**.

   In the custom action page, you can see a list of all the custom action. Each custom action record has associated services and the date modified.

2. Search for your custom action by using any of the following search filters:

   - Enter your custom action name in **Search Custom Action**.
   - Click the show filter menu icon to filter records based on the status of a custom action. The different statuses of a custom action are **Locked** and **Unlocked**.
   - You can also filter the custom action by clicking **Show globally accessible**. You can also include or exclude the globally accessible custom action by checking or unchecking the **Show globally accessible**.
   - You can click **Name** and **Date modified** column headers to sort records.
   - You can select a namespace from the **All namespaces** dropdown list. The list of custom action in the page changes based on the selected namespace. You can select "All namespaces" to see all of the custom action for which you have access.
   - You can use the pagination dropdown to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.

3. Click the custom action for which you want to see the details.

   The name of the custom action and the status of the custom action is displayed. From the overflow menu, you can **Edit**, **Assign access**, **Lock**, **Unlock**, and **Delete** actions. This menu changes as per the status of the custom action.

Editing a custom action

You can edit a custom action from the Manage service library page.

About the task

You can edit locked custom action by changing the status to unlocked. Unlocked custom action can be performed only when the custom action is unassigned from all the service versions that was assigned previously.

Procedure

To edit a custom action, follow these steps:

1. You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.

2. You can select a namespace from the **All namespaces** dropdown list. The list of custom action in the page changes based on the selected namespace. You can also select "All namespaces" to see all the service for which you have access.

3. Edit custom action details. For custom action details, see Creating a custom action.

From the Edit custom actions overflow menu, you can do the following actions based on the status as mentioned below:

- Unlocked: Edit/Assign access/Lock/Delete
- Locked: Edit/Assign access/Unlock/Delete

4. Save the changes.

Assigning a custom action to a service version

Custom actions that are in locked state can only be assigned for a service version.

About the task

Procedure

1. In the navigation bar, click **Library** > **Service Library**. The Manage service library is displayed.
2. In the overflow menu of the service, expand the service and select the service version for which you want to assign the custom action.
3. Click **Assign custom action**.
4. All the custom actions are displayed.

   Note: If the service is assigned to a namespace, then you are displayed with the custom actions that are in the same namespace and global namespace. If the service is assigned to a global namespace, then you see the custom actions of global namespace.
5. Click **Next**
6. Expand the custom actions to add or link the provison action output parameters to the input parameters.
7. You can follow either one of the steps as per the requirement for input parameter.

   a. Click **Add** to add the value (**Note**: Add the correct value depending upon the data type of the input parameters).

   b. Click **Select** and choose from the dropdown list of the interpolated output parameter of the provison action to map the provison action output parameter.
8. Click **Assign**.

Note: If you wish to unassign a particular custom action, then click **Assign custom action** > deselect **Custom action** > click **Next** and **Assign**.

Assigning access to a custom action

Procedure to assign access to multiple namespaces of a custom action.

About the task

Procedure

1. In the navigation bar, click **Library** > **Service Library**. The Manage service library is displayed.
2. Select the overflow menu of the custom action for which you need to assign the access.
3. Click **Assign access**.
4. In the **Assign access** dialog box, select the namespaces. You can select a single or multiple namespaces to which the user has access.
5. Click **Assign**.

Scenarios

- You can set or change a single or multiple valid namespaces to a custom action. The valid namespaces are those which the user has access to.
- A custom action (in locked status) with multiple namespaces can be assigned to service version which has the same namespace associated with it. Global custom actions(in locked status) are available for all the service versions.

Deleting a custom action

You can delete an existing custom action from the Manage service library.

Procedure

1. Go to the Manage service Library page > select the **Custom Action** page.
2. From the overflow menu of a custom action record, select **Delete**.

3. In **Delete** page, enter "delete" and click **Delete**.
4. Custom action gets deleted.

   Notes: Delete action can be performed only when the custom action is not assigned with any service version.

About Managed services Content

The Managed services library comes pre-populated with sample Terraform templates to help you get started quickly. These templates can be deployed across supported clouds that include IBM Cloud, AWS, Azure, GCP, and VMware. The pre-populated sample templates are fully supported by IBM. The Supported Terraform Templates page lists all the supported templates and their repository location in GitHub.

In addition to these pre-populated templates, the GitHub repository contains templates that are provided as-is with no support. You can import these as-is templates using the instructions in Import Template section. If these templates do not satisfy your requirements, then you can fork the repository and change the template to suite your requirements.

The Terraform template content in Managed services library can be classified into starterpack, middleware, and integration contents.

Managed services Starterpack Library

The starterpack library content contains automation building blocks that deploy a select set of cloud-native payloads. Some automated deployments that you can perform with the included templates are:

- MEAN stack
- LAMP stack

The sources for the starterpack library templates are available at GitHub location. You can deploy the starterpack content templates using these instructions.

Managed services Kubernetes cluster templates

The starterpack library content also contains a set of templates to deploy Kubernetes cluster based on native Kubernetes services on public cloud and Import these clusters to the hub cluster.

The GitHub repository also contains as-is templates to deploy Red Hat OpenShift 3.x and 4.2 on VMware and AWS.

Managed services Middleware Library

The middleware templates are available at GitHub location. You can import them using the instructions in Import Template section. Some automated deployments that you can import from this location includes:

- IBM WebSphere Liberty V17
- IBM WebSphere Application Server ND V9
- IBM Db2 V11 or V10.5 stand-alone
- IBM MQ Series V8 or V9

The middleware library templates depend on Chef automation to perform the installation of the middleware products. The chef automation is driven by the Content Runtime environment.

A summary of available middleware templates and associated Chef cookbooks is available at GitHub.

The following links provide more information on the middleware templates.

- Middleware Templates Description: This page details the relevant aspects of Terraform templates used in Middleware library.
- Deploying Middleware Template: This page explains how to deploy a middleware library content.

Managed services Integration Library

Integration library templates provide examples on how to integrate with third party services using Terraform templates. These templates can be wrapped in Service Composer to control the execution.

Integration examples:

- InfoBlox Integraton
- ServiceNow CMDB Configuration Template
- IBM Cloud App Management integration. For more information about IBM Cloud App Management, see IBM Cloud App Management documentation.

Content Runtime Environment

The Content Runtime Environment is set up by deploying the content runtime templates. The following links provide more information on the Content Runtime Environment.

- Content Runtime Environment Overview: This page explains the concepts and architecture of Content Runtime Environment.
- Managing the Content Runtime: This page explains how to set up and manage the Content Runtime Environment.
- Content Runtime in Offline mode: This page details how to install Content Runtime Environment in offline mode (air-gapped system with no-internet).
- Managed services Content APIs - This page details APIs to manage Chef cookbooks and roles.
- Troubleshooting issues and useful links - This page details how to troubleshoot the issues related to Content Runtime environment and Middleware Templates.
- FIPS compliant Content Runtime Environment - This page details how to enable FIPS compliant Content Runtime Environment.

Custom Terraform providers provided by Managed services

CAMC provider

The middleware templates that leverage Chef cookbooks to deploy middleware uses Terraform CAMC provider to perform following Chef related operations:

- Chef bootstrap operation on the target node
- Deploys one or more Chef cookbooks and Chef roles on the target node
- Makes use of the chef-vault gem to store secrets, such as passwords and user names to be defined and used by cookbooks during the software deployment process.

In addition to these operations, Terraform CAMC provider can execute scripts and store the output for use by other resources in your Terraform template.

What's Next

Proceed to Offline Content Installation to know more about offline content template installation and content runtime installation.

Proceed to Starterpack and Integration Content to know more about starterpack and integration library content.

Proceed to Managing the Content Runtime to know more about Content Runtime Environment.

Proceed to Understanding and Deploying Middleware Content Templates to know more about middleware content.

Offline Content Installation

If you do not have internet connectivity while you are installing and configuring Managed services, you can clone the content templates from GitHub and then import them into IBM Cloud Pak for AIOps Managed services library.

IBM Cloud Pak for AIOps provides helper scripts that can be used to clone the content templates and download the prerequisites to setup offline Content Runtime.

To clone the templates, you need internet connection. After cloning the templates, you can:

- Load the cloned templates into your local Guithub Enterprise or GitLab behind the firewall and then import the templates into IBM Cloud Pak for AIOps Managed services

or

- Import the templates directly from cloned file or folder into IBM Cloud Pak for AIOps Managed services.

It is recommended that you load the cloned templates into your local Guithub Enterprise or GitLab and then import from the repository rather than importing them directly from file or folder. If you choose to import directly from file or folder, following helper scripts can be used to bulk import the templates.

- Bulk import starter library
- Bulk import content runtime

The Provisioning an offline Content Runtime provides detailed information on importing content runtime templates and other pre-requisites in an environment with limited or no internet connectivity.

Deploying a starterpack and integration templates

You can deploy the predefined templates in the Starterpacks and Integration Library to your cloud environment including VMware, IBM Cloud, Google Cloud, and AWS.

Deploying Starterpack and Integration Templates Overview

The templates that are pre-loaded during installation, can be deployed using the steps in the Deploying a template. If the template is not loaded you can import the template using the instructions in Importing a template and then deploy them. For an example on how to deploy a starter library template, see Deploying a Single Virtual Machine template.

The integration template ServiceNow CMDB Configuration Template is the only integration template that is pre-loaded. For information on how to deploy this integration template, see Integrating with ServiceNow CMDB.

Starterpack library also contains number of templates to create kubernetes cluster and import them as managed cluster. For information on how to deploy these templates read the following topics

- Deploying Kubernetes cluster
- Deploying OpenShift Container Platform
- Importing Kubernetes target cluster to hub cluster

Validate the deployment

After you deploy the template, you can validate the deployment by checking the logs displayed in the **`Log file`** panel of the deployed instance. You can also check the **`Output variables`** tab of the deployed template and confirm you see the desired output.

Troubleshooting

For details about troubleshooting issues when deploying templates, see Troubleshooting issues and useful links.

Deploying a VMware Single Server template

Deploy the **`SingleVirtualMachine for Terraform 1.x.x`** template in your environment.

Before you begin

VMware templates need to have vmware-tools installed or else the operating system configuration post the deployment is not possible.

Procedure

To deploy the **`SingleVirtualMachine for Terraform 1.x.x`** template, follow these steps:

1. In the navigation bar, click **Library** > **Terraform templates**. The template library is displayed.
2. Click **Starterpacks**.
3. Click **SingleVirtualMachine for Terraform 1.x.x**.
4. Read the information about the template and click **Deploy template**.
5. Enter the required parameters:
   - Specify the **Instance name**.
   - Select the **VMware** cloud connection.
   - Complete **Template parameters**
6. Click **Deploy**. The template instance window is displayed.

In the template instance window, you can see the instance status, the instance information, the related resource details, and the related input and output variables. You can view or copy the instance logs and you can also access the template from which the instance was deployed.

Integrating with ServiceNow CMDB

ServiceNow CMDB Configuration Template allows you to manage the creation and deletion of CMDB records as a part of deploying a Virtual Machine.

You can run this template after the successful execution of a Terraform template that creates a virtual machine. On deployment of this template CMDB Record is created or updated for the virtual machine. On destroy of the template, the CMDB Record is deleted. Ideally you would use this template as an activity in a service composed using service composer. This sample service illusrates how to use this template in a service.

Prerequisites

- A working version of Service Now addressable from the Terraform Engine.

Template Inputs

Variable	Description
cmdb_pass	User to connect to Service Now
cmdb_user	Administrative user password.
cmdb_instance	Target Service Now instance.
cmdb_key	Key value for the Server, this may be the host name. This field relates to the cmdb_ci_server name field.
cmdb_record	A Map of values that constitute the CMDB Record. The structure is user defined and must follow the fields described in the ServiceNow ci_cmdb_server record. The only mandatory value is the name field.

Deploying OpenShift Container Platform on Managed services

There are many as-is OpenShift Container Platform templates available for deployment on Managed services.

The following table lists the templates that are available in Managed services, however, you must import it from Git repository:

Template name in Managed services Library	Description
OpenShift Container Platform Enterprise Installer for OpenShift 3.11	It install enterprise ready HA version of OpenShift Container Platform within your VMware Hypervisor and AWS environment.
OpenShift Container Platform Enterprise Installer for OpenShift 4.2	It install enterprise ready HA version of OpenShift Container Platform within your VMware Hypervisor environment.
OpenShift Container Platform Single Node Installation	It is a deployment wherein all OpenShift nodes are deployed in a single server.
OpenShift modules	OpenShift modules

The supported version of OpenShift Container Platform is 3.11 and 4.2.

Deploying Kubernetes Service cluster on public cloud

You can deploy Kubernetes Service cluster into public clouds, such as IBM Cloud, Amazon Web Services, Microsoft Azure and Google Cloud, and Red Hat OpenShift Kubernetes Service into IBM Cloud using the following templates.

As a first step to integrate, deploy the Kubernetes cluster service with Managed services, create a connection to the Cloud Provider. After you create the connection, you can either deploy one of the following templates or you can create a custom service by using one of the following templates. You can also publish the service in IBM Cloud Pak for AIOps catalog to allow other users a simplified way to create their own clusters.

Template name in Managed services Library	Location and name in Git repository	Description
Managed Kubernetes Service within Microsoft Azure	The `template_kubernetes_aks` Terraform template is available at Git repository.	This template deploys a Kubernetes cluster within Microsoft Azure's Kubernetes Service (AKS).
Managed Kubernetes Service within Google Cloud	The `template_kubernetes_gks` Terraform template is available at Git repository.	This template deploys a Kubernetes cluster within Google's Kubernetes Engine (GKE).
Managed Kubernetes Service within IBM Cloud	The `template_kubernetes_iks` Terraform template is available at Git repository.	This template deploys a Kubernetes cluster within IBM Cloud's Kubernetes Service (IKS).
Managed Kubernetes Service within Amazon Web Services	The `template_kubernetes_eks` Terraform template is available at Git repository.	This template deploys a Kubernetes cluster within Amazon Web Services (EKS).
Red Hat OpenShift single zone cluster on IBM Cloud	The `template_kubernetes_roks` Terraform template is available at Git repository.	The template deploys a Red Hat OpenShift Kubernetes cluster within IBM Cloud (ROKS).

- Refer to Deploying IBM Cloud Kubernetes service template from Managed services for a sample usage on how to use a kubernetes cluster template in a Managed services service.
- Refer to Scaling nodes on IBM Cloud Kubernetes service for a sample usage on how to modify the number of worker nodes to scale up or down.

Working with IBM Cloud Kubernetes Service Template

As a prerequisite to deploying an IBM Cloud Kubernetes Service cluster, verify whether the resource group exists in IBM Cloud and the user ID that is used in IBM Cloud Provider cloud connection has access to this resource group.

As a first step to deploy IBM Cloud Kubernetes Service with Managed services, create a connection to the IBM Cloud Provider. This connection makes use of your IBM Cloud account's API key to communicate between Managed services and IBM Cloud.

After you create the connection, create a custom service by using the Managed Kubernetes Service within IBM Cloud template to deploy an IBM Cloud Kubernetes Service cluster. Deploying IBM Cloud Kubernetes Service template from Managed services describes how to create a custom service from the template. If you choose to add the optional VPN connection, refer to strongSwan Helm Chart on IBM Cloud Pak for AIOps.

After the cluster is created, you can use Deploying helm-tiller template to add Helm and tiller to the cluster.

You can modify the number of nodes in the default pool to scale up and down. For more information, see Scaling nodes on IBM Cloud Kubernetes Service.

Deploying IBM Cloud Kubernetes Service template

Deploy and configure **`IBM Cloud Kubernetes Service for Terraform 1.x.x`** template in Managed services.

Before you begin

- You must have administrator privileges for the deployment of IBM Cloud Kubernetes Service cluster and charts into that cluster. If you do not have the privileges, contact your administrator.
- If you need to initialize helm for clusters that were not created by Managed services, then see Deploying helm-tiller template to add the tiller deployment. For more information about Helm init, see public Helm documentation for Helm init.

Procedure

1. Log in to Managed services.
2. From Managed services menu, go to **Library** > **Services**.
3. In the **Services** pane, click **Create a Service**.
4. Enter your service name and category and click **Create**.
5. Go to the **Composition** tab, drag **`IBM Cloud Kubernetes Service for Terraform 1.x.x`** from the **Templates** > **IBM** section. In Managed services **`IBM Cloud Kubernetes Service for Terraform 1.x.x`** is available in the **Library** > **Terraform templates** page.
6. In the **Basic information** tab > **Connections**, select the IBM Cloud connection from the drop-down list. For more information about IBM Cloud connection, see Configuring an IBM Cloud connection. Use this connection to deploy IBM Cloud Kubernetes Service cluster from Managed services. If you are using a IBM Cloud Kubernetes cluster that is already available in IBM public cloud, see Configuring IBM Cloud Kubernetes Service.
7. Drag any Helm chart based on your requirement.
8. In the **Basic information** tab > **Connection**, pass the output of **`IBM Cloud Kubernetes Service for Terraform 1.x.x`** Terraform template activity in the **Connection** field. The format to provide the value is **`${templates.<activity id>.output.cluster_name`**. For example, **`${templates.ibmcloudcf954ae7.output.cluster_name}`**. The cloud connection is auto-generated.
9. In the **Parameters** tab, select your **namespace**. The values vary based on your connection.

   Note: The **image-repository** is a read-only value. As long as the file is accessible by the cluster, it is usable by the Helm chart. You can deploy a chart that point to images in the public cloud repository. IBM Cloud Kubernetes Service has its own repository of Helm charts. If you want to integrate that, then go to IBM Cloud Pak for AIOps console and add the repository. To configure repositories in IBM Cloud Pak for AIOps, do the following steps:

1. Go to IBM Cloud Pak for AIOps.
2. Click **Manage** > **Helm repositories**. You can see a list of Helm repositories that are configured for IBM Cloud Pak for AIOps. You can click **Sync repositories** to add new repositories. For more information about configuring your repositories in IBM Cloud Pak for AIOps, see IBM Cloud Pak for AIOps IBM Documentation.

10. Save and deploy the service. It creates an IBM Cloud Kubernetes Service cluster and installs Helm tiller on the cluster automatically. This fully integrated and initialized cluster is available for Managed services to deploy Helm charts.

Note: If the following error message is displayed in the activity logs, then ensure that the helm tiller is configured on their IBM Kubernetes Service cluster: **`Error: could not find tiller`**

11. Optionally, when you want to connect workloads in public and private cloud, then deploy VPN Helm charts into the IBM Cloud Kubernetes cluster. You can also create a new service to do the VPN Helm charts deployment and configure a secure connection between the external cluster and IBM Cloud Pak for AIOps.

    Note**: Workaround for strongSwan Helm Chart to work on IBM Cloud Pak for AIOps, see strongSwan Helm Chart on IBM Cloud Pak for AIOps section.

12. Manage clusters from Kubernetes dashboard:

    1. After you deploy the service, go to **Deployed instances** and open your service.
    2. Go to the **Overview** tab > **Activity instances** section.
    3. Click the instance name link. It takes you to the IBM Cloud console where the cluster is running.
    4. Click **Kubernetes dashboard**.
    5. Select your namespace. All of the active workloads are displayed. You can view running pods, discover and perform load balancing, scale deployments, and add replicas from the dashboard.

Deploying helm-tiller template

Deploy helm-tiller template on existing IBM Cloud Kubernetes Service cluster.

Procedure

1. Log in to IBM Cloud: https://console.bluemix.net/containers-kubernetes/clusters.
2. Create a cluster and wait until the cluster is in ready state.
3. Log in to IBM Cloud Pak for AIOps console and click **Automate infrastructure>Managed services**.
4. Create IBM Cloud Kubernetes service connection and connect to the deployed IBM Cloud Kubernetes Service cluster. For the actual steps, see Configuring IBM Cloud Kubernetes Service.
5. Retrieve the IBM Cloud Kubernetes Service cluster and Auth info for the helm-tiller template deployment.
    1. Go to https://console.bluemix.net/containers-kubernetes/clusters.
    2. Follow all the steps there to configure IBM Cloud Kubernetes Service cluster access by using CLI.
    3. Run the export command to export an API key and then log in to bluemix through CLI.

       ```
       export BLUEMIX_API_KEY=<Bluemix Api Key>
       ```

       The output of **`ibmcloud cs cluster-config <cluster name>`** gives the location of your cluster configuration files. For example:

       ```
       export KUBECONFIG=/home/developer/.bluemix/plugins/container-service/clusters/<cluster
       name>/<tiller yml file>
       ```

    4. Go to folder that contains the configuration file. For example:

       ```
       cd /home/developer/.bluemix/plugins/container-service/clusters/<cluster name>
       ```

       There are two files in this folder.

       For example:

       ```
       ca-dal13-jeff-iks-tiller.pem
       kube-config-dal13-jeff-iks-tiller.yml
       ```

    5. In Mac OS, open the following configuration files

       ```
       base64 xxxxxxx.yml > config
       base64 xxxxx.pem > auth
       ```

       In Unix/Linux, open the following configuration files:

```
base64 -w 0 xxxxxx.yml > config
base64 -w 0 xxxxxx.pem > auth
```

When you deploy helm tiller template from starterlibrary, do the following steps:

- Enter the contents of **config** file in the **Cluster configuration** field
- Enter the contents of **auth** file in the **Cluster certificate authority** field

6. Submit deploy and ensure that the deployment completes successfully:

   Run the following command to verify whether the tiller is installed on IBM Cloud Kubernetes Service cluster that is located in the same environment as your configured IBM Cloud Kubernetes Service cluster:

   ```
   kubectl get pods -n kube-system -l app=helm
   ```

   Example output:

   ```
   NAME                             READY     STATUS    RESTARTS   AGE
   tiller-deploy-6cddf6cc56-8hk22   1/1       Running   0          22h
   ```

strongSwan Helm Chart on IBM Cloud Pak for AIOps

Procedure to get strongSwan Helm Chart working on IBM Cloud Pak for AIOps.

Procedure

1. Connect to IBM Cloud Pak for AIOps virtual machine using SSH credentials.
2. Copy the `etcd-secret` as `calico-etcd-secrets` with a command:

   ```
   kubectl -n kube-system get secrets etcd-secret -o json |  jq '.metadata.name = "calico-etcd-secrets"' |
   kubectl -n kube-system  create -f -
   ```

   Note: If you do not have `jq` installed, run the following command to install:

   ```
   apt install jq
   ```

3. Update `calico-config configMap` to add the etcd endpoint details.

   Example snippet to add the etcd_endpoints into calico-config configMap:

   ```
   Get the etcd endpoint details
   ~# etcd_ep=kubectl -n kube-system get cm etcd-config -ojsonpath={.data.etcd_endpoints} -n kube-system;
   echo "$etcd_ep"
   https://9.9.9.9:4001 <===== This is just in our example.
   Execute the following command to edit the calico-config
   kubectl -n kube-system edit cm calico-config
   scroll down in the editor to see the contex
   etcd_ca: /calico-secrets/etcd-ca
   etcd_cert: /calico-secrets/etcd-cert
   etcd_key: /calico-secrets/etcd-key
   ```

   The following is an example when you add `etcd_endpoints`:

   ```
   etcd_ca: /calico-secrets/etcd-ca
   etcd_cert: /calico-secrets/etcd-cert
   etcd_key: /calico-secrets/etcd-key
   etcd_endpoints: https://9.9.9.9:4001 <===== This is just in our example. Use the value specific to your
   env found through $etcd_ep
   ```

4. Save and close the editor.
5. When installing the helm chart, set the parameter `privilegedVpnPod` to true in the config file of strongSwan.

Scaling nodes on IBM Cloud Kubernetes Service

Modify the number of nodes in the default pool to scale up and down.

About this task

Group of Worker nodes with identical configuration are added to a worker pool. A worker pool can be associated with multiple zones. Both single zone and multi-zone clusters are supported. A default zone is automatically created so you do not have to explicitly create one. For this release, only the default worker pool is supported.

Procedure

1. Go to **Library** > **Terraform templates**.
2. Search for **`IBM Cloud Kubernetes Service`** template and deploy it.

   For the steps to deploy the cluster, see Deploying IBM Cloud Kubernetes service template from Managed services.
3. Go to **Deployed instance** > **Terraform templates**.
4. Open the deployed instance of your template.
5. Go to **Modify** tab.
6. Select the version of the template and click **Next**.
7. In the **Additional options** section, select a value in **Number of workers** to scale up the worker nodes.
8. Click **Plan changes**.
9. Check the log output to ensure that the worker node has scaled up or down correctly.
10. In the Apply Changes page, enter apply in **Confirm changes** and click **Apply**.
11. In the **Overview** > **Activity** section, check the status of the plan and apply.

Importing Kubernetes target cluster to hub cluster

You can import Kubernetes clusters from different cloud providers as targets to be managed by Infrastructure management. You can find more information on import in Importing a target managed cluster to the hub cluster.

Managed services provides the following templates to import clusters created on different Kubernetes cloud providers. Managed services provides a set of import templates that can be used on Infrastructure management installed with Red Hat® Advanced Cluster Management for Kubernetes and a set that can be used on Infrastructure management installed without Red Hat Advanced Cluster Management.

These templates are available in Managed services online version automatically. However, you must download it from Git repository for offline version. When you download the templates from the Git repository, ensure you download the right version of the template, as the branch is relative to Infrastructure management version.

Cluster import templates to be used on Infrastructure management installed with Red Hat Advanced Cluster Management

Template name	Location and name in Git repository	Description
Import Microsoft Azure Kubernetes Service into RHACM enabled IBM Cloud Pak for AIOps	The template is available at Git repository	This Terraform template imports an existing Microsoft Azure Kubernetes Service (AKS) Kubernetes cluster into your Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.
Import Amazon EC2 Kubernetes Service into RHACM enabled IBM Cloud Pak for AIOps	The template is available at Git repository	This Terraform template imports an existing Amazon EC2 Kubernetes Service (EKS) Kubernetes cluster into your Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.
Import Google Kubernetes Engine cluster into RHACM enabled IBM Cloud Pak for AIOps	The template is available at Git repository	This Terraform template imports an existing Google Cloud Kubernetes Engine (GKE) Kubernetes cluster into your Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.
Import IBM Cloud Kubernetes Service into RHACM enabled IBM Cloud Pak for AIOps	The template is available at Git repository.	This Terraform template imports an existing IBM Cloud Kubernetes Service (IKS) Kubernetes cluster into your Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.
Import Red Hat OpenShift Kubernetes Service on IBM Cloud into RHACM enabled IBM Cloud Pak for AIOps	The template is available at Git repository.	This Terraform template imports an existing Red Hat OpenShift Kubernetes Service (ROKS) on IBM Cloud Kubernetes cluster into your Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.

Template name	Location and name in Git repository	Description
Import OpenShift Container Platform Kubernetes cluster into RHACM enabled IBM Cloud Pak for AIOps	The template is available at Git repository.	This Terraform template imports an existing Red Hat OpenShift Container Platform (OCP) into your Red Hat Advanced Cluster Management enabled Infrastructure management hub cluster.
Import a Kubernetes cluster into RHACM enabled IBM Cloud Pak for AIOps	The template is available at Git repository	This Terraform template is a generic template that imports an existing Kubernetes cluster into RHACM enabled Infrastructure management.

Note: If you are using offline version of Infrastructure management, then you must use the template from repository template_import_rhacm and branch 5.2.0.

Managing Content Runtime

Before you can deploy virtual machines with Middleware Content installed and configured, run the following prerequisite steps.

Before starting, ensure that the System Requirements are met.

1. Starting from IBM Cloud Pak® for AIOps 4.2.1 release, terraform engine 0.11.x is not shipped with the product. The content runtime and middleware content requires terraform engine 0.11.x. You must use Create a version steps to first import the 0.11.x terraform engine before you deploy a content runtime.
2. Create a cloud connection. For information about creating a connection, see Managing connections.
3. Create the Content Runtime. For more information, see Provisioning and managing your Content Runtime infrastructure.
4. Set up the Software repository. For more information, see Managing a software repository
5. Check that the Middleware Templates are loaded.
6. Check that the IBM Middleware Cookbooks are loaded. For more information, see Kinfe Command Configuration.
7. The Content Runtime template scripts depend on Python for Operating System checks, so ensure that the Operating System image that is provided in the template has Python.

What to do next

Deploy a Middleware Template.

System requirements

Review the minimum system requirements for setting up and running the middleware Content Runtime within a virtual machine.

- Hardware requirements
- Supported operating systems and platforms
- Network requirements
- Port information for Content Runtime virtual machine

Hardware requirements

You must meet the following hardware minimum requirements:

- 4 CPU cores
- 8 GB memory
- 20 GB disk for Operating System
- 100 GB disk for Software Repository
- 40 GB disk for Docker production configuration (optional). For more information, see Production Docker.
- Content Runtime supports only x86 architecture.

Supported operating systems and platforms

The following operating systems and platforms are supported.

- Ubuntu 18.04 LTS
  - VMware Tools must be enabled in the image for VMWare template.
  - Ubuntu Repos with correct configuration must be enabled in the images.
  - Sudo User and password must exist and be allowed for use.

    - Firewall (through iptables) must be disabled.
    - SELinux must be disabled.
    - The system umask value must be set to 0022.
- Red Hat Enterprise Linux (RHEL) 7 (requires Docker-EE)
    - Needs RHEL 7.x image to be created, at least.
    - VMware Tools must be enabled in the image.
    - RHEL Satellite Repos with correct configuration must be enabled in the images.
        - Red Hat Enterprise Linux 7 Server (RPMs)
        - Red Hat Enterprise Linux 7 Server - RH Common (RPMs)
        - Red Hat Enterprise Linux 7 Server - Optional (RPMs)
        - Red Hat Enterprise Linux 7 Server - Supplementary (RPMs)
        - Red Hat Enterprise Linux 7 Server - Extras (RPMs)
    - Sudo user and password must exist and be allowed for use.
    - Firewall (through iptables/firewalld) is disabled.
    - SELinux must be disabled.
    - The system umask value must be set to 0022.
- CentOS 7
    - VMware Tools must be enabled in the image.
    - Sudo user and password must exist and be allowed for use.
    - Firewall (through iptables/firewalld) is disabled.
    - SELinux must be disabled.
    - The system umask value must be set to 0022.

Network requirements

- Static IPv4 address is assigned to each VM. No DHCP is used.
- A static IPv4 address is allocated and reserved prior to its usage.
- The IPv4 address is an IP in a private network that has egress-only access to Internet.
- In addition to IPv4 address, the Gateway, Prefix Length, DNS Suffixes, and DNS Servers need to be obtained prior to the deployment.
- Managed services instance is deployed in the same private network, so deploying from Managed services Instance to VMware environment is truly through private networking.
- The virtual machine must have a connection to the internet for the duration of the configuration process.

Port information for Content Runtime virtual machine

The following table lists the suggested settings for network and firewall:

Table 1. Port Information for Content Runtime

Network Environment	Private IP Only	Port Details
Private	Yes	Allow the following access for all sources on private interface: - SSH - ICMP - HTTP - HTTPS - TCP 8888 (non-secure software repository) - TCP 9999 (secure software repository) - TCP 5443 (pattern manager port)

Chef Versions

The following Chef Server and Client versions have been validated to be used for IBM provided cookbooks within CAM.

- Chef Server : 12.18.14
- Chef Client : 14.1.1

Docker and Docker Compose Versions

- Docker latest version, at the time of this document is : 20.10
- Docker Compose version, 1.11.2

VMWare ESXi Configuration

On an ESX/ESXi host, enable Network Time Protocol (NTP) Client, configure NTP Server, and ensure that the NTP Service is running.

Managed services Content

The Content Runtime environments consist of three key components: a pattern manager, a Chef server, and a software repository. These runtime environments run in your location and space on a single virtual machine (VM). You can have more than one runtime environment to serve different purposes, such as development, test, and production. There must be bidirectional network access between these runtime environments and the Managed services environment.

These advanced Content Runtime environments must be set up before you can deploy a pattern or template. The setup process is a one time step which is performed through the Managed services user interface by using Terraform templates.

As part of the Managed services installation, there is a Terraform Provider container that must have network access to communicate with the VM and network in which the runtime environments operate. It must also be able to communicate with the managed VMs. This container can communicate with the Content Runtime environments over either the public network or a virtual private network (VPN). If communication takes place over a private network, which can be either Amazon or IBM Cloud, the container must also include VPN services. The following Figure 1 illustrates this Managed services Content overview.

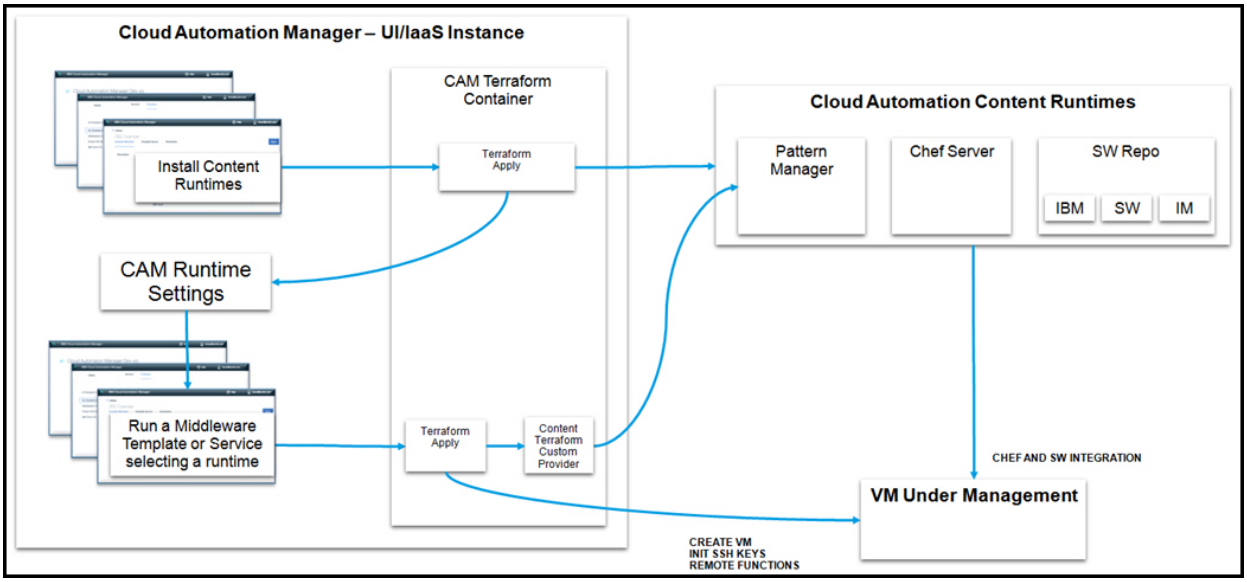


Figure. Managed services Content overview

For optimal performance, the runtime environments should be set up in a location that is geographically close to where you want to deploy your templates. After you configure a runtime environment, the settings are stored in a secure location in Managed services that identifies the deployed **`RUNTIME`** name and the associated settings, such as data center, IP address, and credentials. The settings are later injected as an input to a template during deployment when you select the **`RUNTIME`** name to use to deploy a template. All variables must be modeled in **`camvariables`** of the template that is being deployed to ensure the expected behavior during deployment. When you deploy a template, you select the target runtime environment and the system injects the associated values for that runtime environment into the pattern or template.

Important notes

- Consider where (for example, the region, the data center, and the environment) you want to deploy templates as you plan the setup of your runtime environment.
- Be sure to have a well-defined plan of your desired topology, taking into account available network communication.
- Be sure that the outputs that are stored from the runtime environment align with the expected inputs for your templates. If these outputs and inputs are not aligned, you are prompted to define the runtime parameters for each deployment, which can be difficult and error prone.

Managed services Content frameworks

The Managed services Content Middleware Template Library was built around a set of non-proprietary frameworks which are openly available.

Terraform and Chef were chosen for standardizing the IBM-created content due to the following key advantages:

- Popularity and growth with users.
- A community Model for development, support, and third-party tools.
- They are proven for Enterprises at scale.
- Support for hybrid environments (environments that are based both in the cloud and on-premises).
- Consistent automation format and management for IBM content.

Both communities provide learning resources:

- Terraform: https://www.terraform.io/intro/index.html
- Chef: https://learn.chef.io/#/

Note: Although the IBM Content is created using Terraform and Chef, Managed services does not preclude the use of other popular tools.

What to do next

The topics and tasks in this section guide you through configuring these runtime environments and deploying templates from the Middleware Template Content library. For more information, see Provisioning and managing the Content Runtime.

Enabling FIPS on Pattern Manager and Software Repository containers deployed by Advanced Content Runtime

FIPS is supported on Pattern Manager and Software Repository container deployed by Advanced Content Runtime starting from Advanced Content Runtime template version 2.5.

To enable FIPS, when you deploy the Advanced Content Runtime, set the value of attributes **Enable FIPS on pattern manager** and **software repository containers** to True. It configures the Pattern Manager and Software Repository container to operate in FIPS mode. After you deploy the template, you cannot change the FIPS configuration on Pattern Manager and Software Repository container. To enable FIPS for Chef Server or Chef Client, you must provision the Chef Server and middleware templates that use chef client on a Red Hat 6 or 7 system. Chef do not support FIPS on Ubuntu. For more details on Chef FIPS support, see Chef documentation.

Provisioning and managing your Content Runtime infrastructure

Complete the following steps to configure your Content Runtime environment.

About this task

The Content Runtime is a virtual machine. A Chef server is installed on the virtual machine as part of the configuration of the Content Runtime. Two docker containers are installed on the machine to help communicate with the Chef server and to manage the software repository.

There are four templates:

- VMware
- Other
- IBM
- Amazon

The **`VMware`** template deploys a virtual machine to the target VMware vSphere cloud. The **`Other`** template deploys the Content Runtime to an existing virtual machine. This allows you to instantiate a virtual machine and perform as much or as little configuration as needed to create the Content Runtime. IBM and Amazon templates use a cloud provided Ubuntu 14.04 image as a deployment virtual machine.

The configuration process installs the following components on the Content Runtime virtual machine:

- Chef server
- Docker runtime
- Pattern manager docker container
- Software repository docker container

These components are illustrated in Figure 2. Components on the Content Runtime virtual machine.

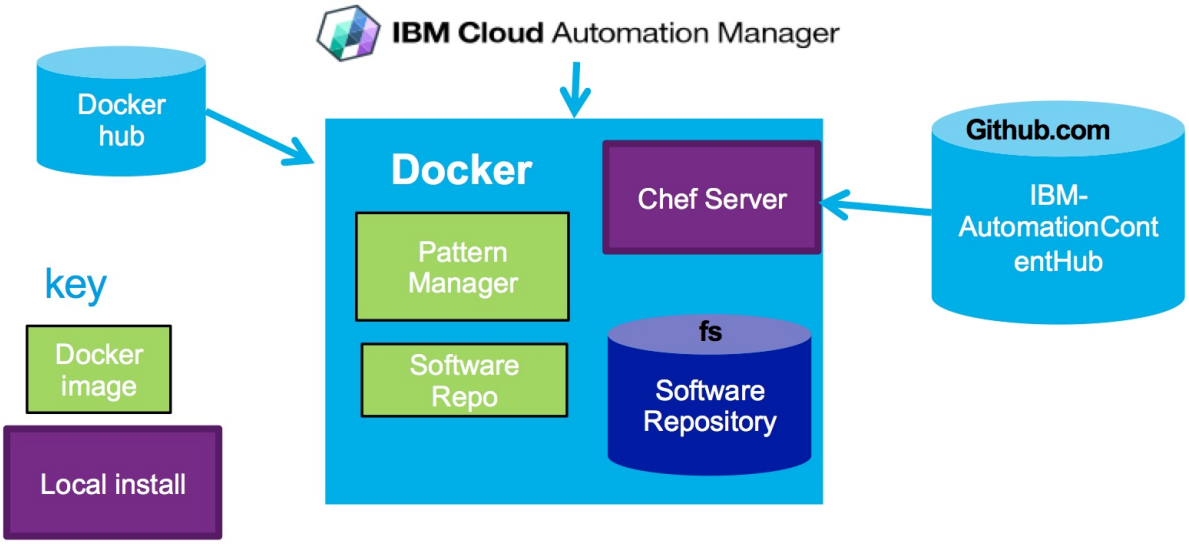


Figure. Components on the Content Runtime virtual machine

After you complete the task, you must populate the software repository on the file system of the Content Runtime virtual machine. See Populating the Software Repository and Populating the Install Manager Repository for more details.

Selecting a template

Before selecting a template, consider the following:

- VMware

  Beside the common parameters listed in the following section, you must have all the information needed to instantiate a virtual machine through the VMware vCenter. This information is entered in the Content Runtime template. The parameters are listed in the following Common Parameter Values section.

- Amazon Web Services

  When deploying an AWS EC2 instance, information such as the region (for example **`us-east-1`**), a VPC subnet ID and a security group ID will be required besides the common parameters. A private IP address associated to the provided subnet is optional.

  Note When providing a **`Key Name for Pattern Manager key set`**, the key needs to be created manually in the AWS EC2 dashboard. This can be done by accessing the **Key Pairs** page. Click **Import Key Pair** and provide the public key associated to the Pattern Manager. Selecting **Create Key Pair** will generate a key which is not supported by the system.

- IBM Cloud

  An IBM Datacenter, such as **`dal09`** for Dallas 09, will be required alongside hardware specifications. The default values for these specifications are the minimum recommended for a Content Runtime deployment, but they can be tweaked. Optional settings include the ability to associate a portable IP address to the virtual machine.

- Other

  Beside the common parameters, the **`Other`** template allows you to provide the IP address of an existing virtual machine and login credentials to the Content Runtime template. An existing VMware vCenter, Amazon Web Services or IBM Cloud account can be used to manually create a virtual machine to be used. This may be the easiest solution for first time users.

Input parameter

For more information about public and private keys, see SSH Keys and related input required while deploying content runtime. Deploying a content runtime requires the following categories of parameters:

- For the cloud connection name, see Managing connections
- Connection information for the virtual machine: operating system credentials for the template
- New passwords and passphrases for the software repository content and pattern manager REST services
- SSH public and private keys for the pattern manager. It can be generated with the following commands:
  - In Mac OS:

    ```
    ssh-keygen -t rsa -f [output_keyfile] -N "" -C "";base64 [output_keyfile] > [output_keyfile].enc
    ```

- In Unix:

  ```
  ssh-keygen -t rsa -f [output_keyfile] -N "" -C ""; base64 -w 0 [output_keyfile] > [output_keyfile].enc
  ```

  The [output_keyfile].enc output file is the encoded private key used. The [output_keyfile].pub file is the public key. The content of these files are used as input to the template.

  Note:

  - Pattern manager can only communicate with Linux guests. Even if cygwin is present, the Windows operating system is not supported.
  - For AWS and IBM Cloud, the generated public key must be imported as an SSH key pair. The name of this key pair will be used for part of the deploy process.
  - The pattern manager SSH public and private key must be a RSA key. The ED25519 algorithm (OpenSSH key) is not supported.

Parameter values which are grayed out are immutable values that are displayed as part of the Content Runtime template for informational value only.

The virtual machine must have a connection to the internet for the duration of the configuration process.

Template variables

The following tables contain a list of values which are required for creating the Content Runtime. In the first table, there are the values which are common for all providers followed by a table for specific values for each cloud provider.

- VMware Content Runtime template variables
- Other Content Runtime template variables
- IBM Content Runtime template variables
- Amazon Content Runtime template variables

Procedure

1. Start the creation process by displaying the menu in the Managed services user interface.
2. Select **Manage** > **Content runtimes**.
3. Click **Create content runtime**.
4. Select a namespace value from the **Namespace** drop-down list.
5. Enter a name for your Content Runtime. This name is used for selecting environment value to inject, so the name should be descriptive of the environment.
6. Select the Cloud Provider.
7. Select a value from the **Runtime** drop-down list. The values populated in the list are dependent on your cloud provider.
8. Click **Create**.
9. Select a cloud connection from the **Cloud connection** drop-down list.
10. Enter the remaining values for your Cloud Provider.
11. Click **Deploy**.

Result

- A virtual machine is created. The Content Runtime is configured on the virtual machine.
- Two disks are created:
  - 100 GB disk with mount point `/opt/ibm/docker/software-repo`
  - 25 GB root/boot disk
  - Optional: 40 GB Docker `direct-lvm` mode (only supported in the `Other` template, see Production Docker for more information)
- Docker-compose is installed on the machine to manage Docker images.
- Docker runs on the machine, with the following containers:
  - Pattern manager Docker container
  - Software Repository Docker container
- Cookbooks are pulled from GitHub and uploaded to the Chef server.
- When you deploy a new advanced content runtime, a data object containing it's meta-data is created. The name of the data object is same as the content runtime name and is of type `advanced_content_runtime_chef` Data Type. The namespace of the data object is same as the namespace of the content runtime. Optionally, if you want this data object to be available to middleware template deployments that are in a different namespaces, then make this data object as globally accessible.

If any problem occurs, see Troubleshooting and known issues.

Search for content runtimes

In **Content runtimes** page, you can use the following methods to search for content runtimes:

- When you want to view **Content runtimes** of a particular namespace, select a namespace from the drop-down list. The list of content runtimes in the page changes based on the selected namespace. You can also select "All namespaces" to see all content runtimes for which you have access.
- You can enter text in **Search items** to filter the content runtime list.
- You can use the show filter icon next to search text to filter records based on provider type.

Production Docker

Docker recommends configuring direct lvm mode for production environments.

The following two methods can be used to allow the Content Runtime to configure production docker:

1. **`Other`** template
    - Define a third disk of 40 GB before configuring the Content Runtime virtual machine. This method only works on machines that are not already configured.
2. On an existing Content Runtime virtual machine.
    - Add a 40 GB disk.
    - Delete the **`daemon.json`** with the command **`sudo rm /etc/docker/daemon.json`**.
    - Execute the command **`~/advanced-content-runtime/launch-docker-compose.sh`**.

Note: There is an underlying option missing on the **`lvcreate`** command on Ubuntu 14.04, therefore the Content Runtime does not attempt to configure **`direct-lvm`** on Ubuntu 14.04.

What to do next

- After the Content Runtime is deployed, you can log in to the Content Runtime virtual machine either by using the password or by using the SSH key.

  VMWare Cloud

  To log in to the Content Runtime deployed on VMWare, use the password if you had provided one during the deployment or use the SSH private key that is associated with the SSH public key. This SSH public key is the value you provided for input *User's Public Key*.

  IBM Cloud

  To log in to the content runtime deployed on IBM Cloud, do the following steps:

  1. Click the **Output variables** link in the Content Runtime deployed instance details page.
  2. Copy the value of Private Key for Content Runtime virtual machine by using the copy icon.
  3. Save the copied contents to a file and use this private key to SSH into the virtual machine. Alternatively, if you had provided a value for *User's Public Key*, then you can use the private key associated with this value.

  Amazon Cloud

  To log in to the content runtime deployed on IBM Cloud, do the following steps:

  1. Click the Output Variables link in the Content Runtime deployed instance details page.
  2. Copy the value of Private Key using the copy icon.
  3. Save the copied contents to a file and use this private key to SSH into the virtual machine. Alternatively, if you had provided a value for *User's Public Key*, then you can use the private key associated with this value.
- You must complete the Software Repository configuration step as described in Software repository configuration.

IBM Content Runtime template variables

The following tables list the IBM template variables.

Cloud Provider

The following table lists the parameters for a cloud provider.

Parameter	Default Value	Description
Virtual Machine's domain name		Virtual Machine's domain name for the Content Runtime Virtual Machine.

Parameter	Default Value	Description
Public Key Name	cam_public_key	Provide a public key name. The key will be created and stored in the IBM Cloud as a resource with the name Hostname_PrivateKeyName. The private portion of the key can be found in the log file in the Terraform data.
OS Image Name	UBUNTU_16_64	An operating system reference code that is used to provision the computing instance.
Data Center		Specify the data center to create the Content Runtime virtual machine.
Private subnet		Private subnet for the private network interface of the instance. Accepted values are primary private networks and can be found in the subnets doc.
Private VLAN name		Private VLAN name for the network interface of the instance.
Dedicated Account Host		Specify whether or not the instance must only run on hosts with instances from the same account
Billing Type		Specify the billing type for the instance. The computing instance can be billed based on an hourly or monthly usage basis.
Network Connection Speed		Specify the connection speed for the instance's network components.
CPU cores		The number of CPU cores to allocate. The default value is sufficient for the Content Runtime.
RAM		The amount of RAM to allocate for the virtual machine. The default value is sufficient for the Content Runtime.

Cloud Provider Optional Settings

The following table lists the optional settings parameters for a cloud provider.

Parameter	Default Value	Description
Portable private IP address		Portable private IPv4 address to be assigned to this network interface for the Content Runtime virtual machine. If provided, the address needs to be static.
Portable private netmask		Portable private netmask for the private IP added to the new Content Runtime virtual machine. Required when assigning a portable IP address.
Portable private gateway		Portable private gateway address to be assigned to the new Content Runtime virtual machine. Required when assigning a portable IP address.

Additional Settings

The following table lists the additional settings parameters.

Parameter	Default Value	Description
Network Visibility		The network connection associated with the Content Runtime virtual machine instance. If network visibility is private, connection must exist from the IBM Cloud Pak for AIOps to the network connection associated with the Content Runtime virtual machine being created.

Managed services Parameters

The following table lists the parameters for Managed services.

Parameter	Default Value	Description
Enable FIPS on pattern manager and software repository containers	False	True to enable FIPS on pattern manager and software repository container.
Pattern Manager Access Passphrase		Passphrase used to access the REST API calls to the Pattern Manager.
Pattern Manager Administration Passphrase		Administration passphrase that is used to access the administrative REST API calls to the Pattern Manager.
Key Name for Pattern Manager key set		For Amazon and IBM clouds, the key must exist as a resource. For other provider types, the content is simply the name associated with the key.
Private Key for Pattern Manager - Base 64 encoded		Private key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must be base64 encoded and must be associated with the public key named by the Key Name for Pattern Manager key set.
Public Key for Pattern Manager		Public key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must match the public key named by the Key Name for Pattern Manager key set.

Parameter	Default Value	Description
User's Public Key		Public key that is added to the Content Runtime virtual machine's `authorized_keys` file so as to allow access using the associated private key.
Content Runtime Host Name	ibm-content-runtime	The host name of the virtual machine being created to be used as the Content Runtime. The name is just the host name and not the fully qualified host name.

Note: For more information about generating public and private keys, see SSH Keys and related input required while deploying content runtime.

Docker & Chef Settings

The following table lists the parameters for Docker and Chef settings.

Parameter	Default Value	Description
Docker Registry	hub.docker.com	Docker registry to pull the Content Runtime Docker images. The images by default reside on the public Docker hub. The value is read only. It provides you the location where an image is pulled.
Chef License Agreement		Newer Chef Server, Client, Chef DK require license agreement with Chef for commercial. By typing Agree to this field you confirm that you are either using old chef version or you have the required license agreement.
ChefDK version	3.0.36	Stable version of Chef DK that will be installed on pattern manager container.
Docker Enterprise Edition Repository URL		This is an optional parameter. It is mandatory only if you use RHEL as the base image without Docker installed. RHEL supports Docker Enterprise Edition only. After you purchase the license, get the Docker Enterprise Edition Repository URL from https://store.docker.com/my-content. Docker Community Edition is used by default on other supported operating system.
Chef server Organization Name	chef-org	Chef server Organization name that is associated with the configuration of the Chef server.
Chef Administrator User name	chef-admin	Chef administrator user name that is associated with the configuration of the Chef server.
Chef Server version	12.11.1	Stable version of Chef server to install

Software Repository Settings

The following table lists the parameters for software repository settings.

Parameter	Default Value	Description
Software Repository User name	repouser	User name to access the Software Repository.
Password for Software Repository		Password for Software Repository User. The combination of user name and password is required to access the Software Repository.
Software Repository Port Number	8888	Port number for Software Repository HTTP access.
Software Repository Secure Port Number	9999	Port number for Software Repository HTTPS access.

Amazon Web Services Content Runtime template variables

The following tables list the Amazon template variables.

Cloud Provider

Parameter	Default Value	Description
AWS User ID	ubuntu	An internal field that is needed wherever an ID is required for an image.
Security Group ID		The ID of security group that exists in Amazon cloud. The security group is used while creating the Content Runtime virtual machine.
VPC Subnet ID		The VPC Subnet ID to use while creating Content Runtime. The subnet must exist in Amazon.
Amazon EC2 Instance Type	m3.large	The type of the Amazon EC2 Instance. The default setting is the recommended size for running the Content Runtime.
AWS region		The AWS region to create Content Runtime virtual machine

Cloud Provider Optional Settings

Parameter	Default Value	Description
Private IP Address		Static IPv4 address to be assigned to Content Runtime virtual machine. The provided IP address must be associated to the subnet id. If empty, AWS assigns an IP address that is associated to the subnet id.

Additional Settings

Parameter	Default Value	Description
Network Visibility		The network connection that is associated with the Content Runtime virtual machine instance. If network visibility is private, connection must exist between IBM Cloud Pak for AIOps and the network connection. The network connection is associated with the Content Runtime virtual machine that is being created.

Managed services Parameters

Parameter	Default Value	Description
Enable FIPS on pattern manager and software repository containers	False	True to enable FIPS on pattern manager and software repository container.
Pattern Manager Access Passphrase		Passphrase used to access the REST API calls to the Pattern Manager.
Pattern Manager Administration Passphrase		Administration passphrase that is used to access the administrative REST API calls to the Pattern Manager.
Key Name for Pattern Manager key set		For Amazon cloud, the key must exist as a resource. For other provider types, the content is simply the name associated with the key.
Private Key for Pattern Manager - Base 64 encoded		Private key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must be base64 encoded and must be associated with the public key named by the Key Name for Pattern Manager key set.
Public Key for Pattern Manager		Public key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must match the public key named by the Key Name for Pattern Manager key set.
User's Public Key		Public key that is added to the Content Runtime virtual machine's `authorized_keys` file so as to allow access using the associated private key.
Content Runtime Host Name	ibm-content-runtime	The host name of the virtual machine being created to be used as the Content Runtime. The name is just the host name and not the fully qualified host name.

Note: For more information about generating public and private keys, see SSH Keys and related input required while deploying content runtime.

Docker & Chef Settings

Parameter	Default Value	Description
Docker Registry	hub.docker.com	Docker registry to pull the Content Runtime Docker images. The images by default reside on the public Docker hub. The value is read only. It provides you the location where an image is pulled.
Chef License Agreement		Newer Chef Server, Client, Chef DK require license agreement with Chef for commercial. By typing Agree to this field you confirm that you are either using old chef version or you have the required license agreement.
ChefDK version	3.0.36	Stable version of Chef DK that will be installed on pattern manager container.
Docker Enterprise Edition Repository URL		This is an optional parameter. It is mandatory only if you use RHEL as the base image without Docker installed. RHEL supports Docker Enterprise Edition only. After you purchase the license, get the Docker Enterprise Edition Repository URL from https://store.docker.com/my-content. Docker Community Edition is used by default on other supported operating system.
Chef server Organization Name	chef-org	Chef server Organization name that is associated with the configuration of the Chef server.
Chef Administrator User name	chef-admin	Chef administrator user name that is associated with the configuration of the Chef server.
Chef Server version	12.11.1	Stable version of Chef server to install

Software Repository Settings

Parameter	Default Value	Description

Parameter	Default Value	Description
Software Repository User name	repouser	User name to access the Software Repository.
Password for Software Repository		Password for Software Repository User. The combination of user name and password is required to access the Software Repository.
Software Repository Port Number	8888	Port number for Software Repository HTTP access.
Software Repository Secure Port Number	9999	Port number for Software Repository HTTPS access.

VMware Content Runtime template variables

The following tables list the VMware template variables.

Cloud Provider

Parameter	Default Value	Description
vSphere Datacenter		The name of a Datacenter in which to launch the virtual machine.
vSphere Disk Datastore		Datastore for the created disks. Two disk are created, one for the OS template, the other for the data including the software repository content.
vSphere Default Resource Pool		Name of the default resource pool for the cluster. Specified as 'cluster_name/resource_pool'
vSphere Network Adapter Type	vmxnet3	Network interface type
Domain name		Content Runtime Virtual Machine's domain name
CPU cores		The number of CPU cores to allocate for the Content Runtime. The default value is sufficient for the Content Runtime.
RAM		The amount of RAM to allocate for the virtual machine. The default value is sufficient for the Content Runtime.
OS Template Name		The folder/template name to use to configure the OS on the Content Runtime Virtual Machine.
Size of image disk (in GBs)	25	The size of the disk for the Content Runtime. It must be equal or larger than the provided image
Size of data disk (in GBs)	100	The size of the data disk for the Content Runtime, and Software Repository.
Network interface label		Label to assign to this network interface.
Static IPv4		Static IPv4 address to be assigned to this network interface for the Content Runtime virtual machine. The address must be a static IP.
Gateway IP address		IPv4 gateway address to be used.
IPv4 prefix length (CIDR)	24	Integer value between 1 and 32 for the prefix length, CIDR, to use when statically assigning an IPv4 address
Template image's user name		The user name to use while configuring the Content Runtime virtual machine.
Template image's password		The password to use for the provided SSH user name while configuring the Content Runtime virtual machine. The password is optional if an SSH key is provided.
Template image's private key for SSH connection - Base 64 encoded		A base 64 encoded SSH private key used for configuring the Content Runtime virtual machine. This key is optional if a password is specified.

Cloud Provider Optional Settings

Parameter	Default Value	Description
vSphere Folder		Name of the vSphere folder where the virtual machine is created.
Keep Disks On Remove	false	Should disks be kept on vSphere even if the virtual machine is destroyed?
DNS servers		List of DNS servers for the virtual network adapter. For more than one DNS, the values can be entered by adding more fields.
DNS suffix list		List of DNS suffixes for the virtual network adapter. For more than one suffix, the values can be entered by adding more fields.

Additional Settings

Parameter	Default Value	Description
Network Visibility	public	The network connection associated with the Content Runtime virtual machine instance. If network visibility is private, connection must exist from the IBM Cloud Pak for AIOps to the network connection associated with the Content Runtime virtual machine being created.
Prerequisite checker strictness		If set to strict, the prerequisite checker stops execution if one or more requirements is not installed. If set to lenient, If set to lenient, the prerequisite checker proceeds to install any missing requirement.

Managed services Parameters

Parameter	Default Value	Description
Enable FIPS on pattern manager and software repository containers	False	True to enable FIPS on pattern manager and software repository container.
Pattern Manager Access Passphrase		Passphrase used to access the REST API calls to the Pattern Manager.
Pattern Manager Administration Passphrase		Administration passphrase that is used to access the administrative REST API calls to the Pattern Manager.
Key Name for Pattern Manager key set		For Amazon and IBM clouds, the key must exist as a resource. For other provider types, the content is simply the name associated with the key.
Private Key for Pattern Manager - Base 64 encoded		Private key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must be base64 encoded and must be associated with the public key named by the Key Name for Pattern Manager key set.
Public Key for Pattern Manager		Public key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must match the public key named by the Key Name for Pattern Manager key set.
User's Public Key		Public key that is added to the Content Runtime virtual machine's `authorized_keys` file so as to allow access using the associated private key.
Content Runtime Host Name	ibm-content-runtime	The host name of the virtual machine being created to be used as the Content Runtime. The name is just the host name and not the fully qualified host name.

Note: For more information about generating public and private keys, see SSH Keys and related input required while deploying content runtime.

Docker & Chef Settings

Parameter	Default Value	Description
Docker Registry	hub.docker.com	Docker registry to pull the Content Runtime Docker images. The images by default reside on the public Docker hub. The value is read only. It provides you the location where an image is pulled.
Chef License Agreement		Newer Chef Server, Client, Chef DK require license agreement with Chef for commercial. By typing Agree to this field you confirm that you are either using old chef version or you have the required license agreement.
ChefDK version	3.0.36	Stable version of Chef DK that will be installed on pattern manager container.
Docker Enterprise Edition Repository URL		This is an optional parameter. It is mandatory only if you use RHEL as the base image without Docker installed. RHEL supports Docker Enterprise Edition only. After you purchase the license, get the Docker Enterprise Edition Repository URL from https://store.docker.com/my-content. Docker Community Edition is used by default on other supported operating system.
Chef server Organization Name	chef-org	Chef server Organization name that is associated with the configuration of the Chef server.
Chef Administrator User name	chef-admin	Chef administrator user name that is associated with the configuration of the Chef server.
Chef Server version	12.11.1	Stable version of Chef server to install

Software Repository Settings

Parameter	Default Value	Description
Software Repository User name	repouser	User name to access the Software Repository.
Password for Software Repository		Password for Software Repository User. The combination of user name and password is required to access the Software Repository.
Software Repository Port Number	8888	Port number for Software Repository HTTP access.

Parameter	Default Value	Description
Software Repository Secure Port Number	9999	Port number for Software Repository HTTPS access.

Other Content Runtime template variables

The following tables list the Other template variables.

Cloud Provider

Parameter	Default Value	Description
Static IPv4 address		Static IPv4 to assign to this network interface. This must be a static IP, and it is the IP address of the machine as known to the network.
Virtual Machine's Domain Name		Content Runtime Virtual Machine's domain name
Operating System User name		The user name which can be used to sign on to the system to configure the machine as a Content Runtime.
Operating System password		Password for signing onto the system to configure the machine as a Content Runtime. If an SSH key is provided this parameter is not needed.
OS SSH Private Key - Base64 encoded		The SSH key to connect to the virtual machine to allow for configuration of the Content Runtime. The key is optional if the password is provided for the operating system user name. The key is base64 encoded.

Additional Settings

Parameter	Default Value	Description
Network Visibility	public	The network connection associated with the Content Runtime virtual machine instance. If network visibility is private, connection must exist from the IBM Cloud Pak for AIOps to the network connection associated with the Content Runtime virtual machine being created.
Prerequisite checker strictness		If set to strict, the prerequisite checker stops execution if one or more requirements is not installed. If set to lenient, the prerequisite checker proceeds to install any missing requirement.
File system encryption passphrase		Passphrase for encrypted drives.

Managed services Parameters

Parameter	Default Value	Description
Enable FIPS on pattern manager and software repository containers	False	True to enable FIPS on pattern manager and software repository container.
Pattern Manager Access Passphrase		Passphrase used to access the REST API calls to the Pattern Manager.
Pattern Manager Administration Passphrase		Administration passphrase that is used to access the administrative REST API calls to the Pattern Manager.
Key Name for Pattern Manager key set		For Amazon and IBM clouds, the key must exist as a resource. For other provider types, the content is simply the name associated with the key.
Private Key for Pattern Manager - Base 64 encoded		Private key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must be base64 encoded and must be associated with the public key named by the Key Name for Pattern Manager key set.
Public Key for Pattern Manager		Public key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must match the public key named by the Key Name for Pattern Manager key set.
User's Public Key		Public key that is added to the Content Runtime virtual machine's `authorized_keys` file so as to allow access using the associated private key.

Note: For more information about generating public and private keys, see SSH Keys and related input required while deploying content runtime.

Docker & Chef Settings

Parameter	Default Value	Description
Chef Client Installation Binaries Path		Provide the local path to chef client installation binaries. The format must be `/path/to/chef-clients` without any trailing slash. This value is required for offline install.
Docker Registry	hub.docker.com	Docker registry to pull the Content Runtime Docker images. The images by default reside on the public Docker hub. The value is read only. It provides you the location where an image is pulled.
Chef License Agreement		Newer Chef Server, Client, Chef DK require license agreement with Chef for commercial. By typing Agree to this field you confirm that you are either using old chef version or you have the required license agreement.
ChefDK Version	3.0.36	Stable version of Chef DK that will be installed on pattern manager container.
Docker Enterprise Edition Repository URL		This is an optional parameter. It is mandatory only if you use RHEL as the base image without Docker installed. RHEL supports Docker Enterprise Edition only. After you purchase the license, get the Docker Enterprise Edition Repository URL from https://store.docker.com/my-content. Docker Community Edition is used by default on other supported operating system.
Chef server Organization Name	chef-org	Chef server Organization name that is associated with the configuration of the Chef server.
Chef Administrator User name	chef-admin	Chef administrator user name that is associated with the configuration of the Chef server.

Optional Settings for Offline Installation

Parameter	Default Value	Description
Docker installation file		Optional file location of a docker .deb/.rpm file to be installed on the system. If the installation package has been stored within the content runtime system, for example, docker.deb saved to /tmp, then the input would be file:///tmp/docker.deb. If the installation package is stored in a server with HTTP access, then the input would be http://<server_IP>/path/to/docker.deb.
Docker Compose installation file		Optional file location of a Docker Compose executable file to be installed on the system. If the installation package has been stored within the content runtime system, for example, docker-compose saved to /tmp, then the input would be file:///tmp/docker-compose. If the installation package is stored in a server with HTTP access, then the input would be http://<server_IP>/path/to/docker-compose.
Software Repository Docker image location		Optional file location of the Software Repository Docker image to be loaded to the system. If the installation package has been stored within the content runtime system, for example, camc-sw-repo saved to /tmp, then the input would be file:///tmp/camc-sw-repo. If the installation package is stored in a server with HTTP access, then the input would be http://<server_IP>/path/to/camc-sw-repo.
Pattern Manager Docker image location		Optional file location of the Pattern Manager Docker image to be loaded to the system. If the installation package has been stored within the content runtime system, for example, camc-pattern-manager saved to /tmp, then the input would be file:///tmp/camc-pattern-manager. If the installation package is stored in a server with HTTP access, then the input would be http://<server_IP>/path/to/camc-pattern-manager.
Chef Server Installation File		Binary installation file for Chef server. Note: the installation package is OS dependent. This entry may contain an http://, https://, or file:// link to an installation package (.deb/.rpm). If the installation package has been stored within the content runtime system, for example, chef.rpm saved to /tmp, then the input would be file:///tmp/chef.rpm. If the installation package is stored in a server with HTTP access, then the input would be http://<server_IP>/path/to/chef.rpm.

Software Repository Settings

Parameter	Default Value	Description
Software Repository User name	repouser	User name to access the Software Repository.
Password for Software Repository		Password for Software Repository User. The combination of user name and password is required to access the Software Repository.
Software Repository Port Number	8888	Port number for Software Repository HTTP access.
Software Repository Secure Port Number	9999	Port number for Software Repository HTTPS access.

Using an existing Chef server when deploying a Content Runtime

Using the Content Runtime templates, you can provide an existing Chef server to which the Pattern Manager will connect.

Requirements

To use an existing Chef server when deploying a Content Runtime, the following are needed:

- The Chef server needs to have a static IP address assigned to it.

  Note: This IP address must be accessible from the Content Runtime.

- A user has to be created in the existing Chef server and associated to an organization. For more information on how to perform these steps, see Chef's documentation.

  Note: When creating a user, by default, the **`chef-server-ctl user-create`** command will print the certificate file for the new user to the standard output. To store this result in a file, use the **`-f`** option.

Procedure

To deploy a Content Runtime that will connect to an external Chef server, a custom template needs to be selected.

1. Start the creation process by selecting **Manage** > **Content runtimes** in the Managed services.
2. Click **Create content runtime**.
3. Enter a descriptive name for your Content Runtime.
4. Select the Cloud Provider. (Note: Select Other as cloud provider for offline install).
5. From the dropdown select the option **CAM Content Runtime with existing Chef Server**
6. The additional fields that need to be filled in are:
   - **Chef Server FQDN**: This is the fully qualified domain name of the Chef server.
   - **Chef Server IP Address**: The static IPv4 address of the existing Chef server.
   - **Chef Server Certificate - Base 64 Encoded**: The certificate (usually **`.pem`** format) containing the private key associated to the provided Chef user in the server. This key allows the Pattern Manager to perform requests to the Chef server.
   - **Install Chef Cookbooks** This input represents the option to install the set of cookbooks that are usually bundled on a Content Runtime deployment. If needed, this value can be set to *False* and cookbooks can be loaded manually into the server by following Managed services Content APIs.
7. Click **Deploy**.

Input Parameters

The following tables contain a complete list of all the input parameters available in these templates

- VMware Content Runtime with Existing Chef Server template variables
- Other Content Runtime with Existing Chef Server template variables
- IBM Content Runtime with Existing Chef Server template variables
- Amazon Content Runtime with Existing Chef Server template variables

Obtaining the Chef server FQDN

The fully qualified domain name of the existing Chef server can be obtained from the **`/etc/hosts`** file. It contains the machine's hostname and its associated domain name.

Obtaining the Chef server certificate

When a new user is created with the **`user-create`** option in **`chef-server-ctl`**, its private key is displayed in the standard output. This value can be copied into a file or the **`-f`** option can be used to automatically store it in a specific location.

If the existing Chef server was created using a Content Runtime deployment, then by default the created organization would be named **`chef-org`** and its user **`chef-admin`**. The **`.pem`** file in this case can be found at **`/etc/opscode/chef-admin.pem`**.

Base 64 encoding

To encode the **`.pem`** file into base 64, use the following command in a terminal window:

```
bash
base64 <filename>.pem -w 0
```

The resulting value must be provided in the **Chef Server Certificate - Base 64 Encoded** field.

Verifying Installation

The verification script bundled with the Content Runtime deployment should be able to establish a connection to the external Chef server and obtain the number of existing cookbooks and roles. If this step is successful, the **Logs** section displays the following messages:

```
bash
null_resource.call_launcher (remote-exec): [INFORMATIONAL] An external Chef server was configured on
installation
...
null_resource.call_launcher (remote-exec): [SUCCESS] Chef Cookbooks verified successfully
null_resource.call_launcher (remote-exec): [SUCCESS] Cookbooks response verified successfully
null_resource.call_launcher (remote-exec): [SUCCESS] Total Chef Cookbook count 13
null_resource.call_launcher (remote-exec): [SUCCESS] Total Chef role count 38
```

IBM Content Runtime with Existing Chef Server template variables

The following tables list the IBM template variables.

Cloud Provider

Parameter	Default Value	Description
Virtual Machine's domain name		Virtual Machine's domain name for the Content Runtime Virtual Machine.
Public Key Name	cam_public_key	Provide a public key name. The key will be created and stored in the IBM Cloud as a resource with the name Hostname_PrivateKeyName. The private portion of the key can be found in the log file in the Terraform data.
OS Image Name	UBUNTU_16_64	An operating system reference code that is used to provision the computing instance.
Data Center		Specify the data center to create the Content Runtime virtual machine.
Private subnet		Private subnet for the private network interface of the instance. Accepted values are primary private networks and can be found in the subnets doc.
Private VLAN name		Private VLAN name for the network interface of the instance.
Dedicated Account Host		Specify whether or not the instance must only run on hosts with instances from the same account
Billing Type		Specify the billing type for the instance. The computing instance can be billed based on an hourly or monthly usage basis.
Network Connection Speed		Specify the connection speed for the instance's network components.
CPU cores		The number of CPU cores to allocate. The default value is sufficient for the Content Runtime.
RAM		The amount of RAM to allocate for the virtual machine. The default value is sufficient for the Content Runtime.

Cloud Provider Optional Settings

Parameter	Default Value	Description
Portable private IP address		Portable private IPv4 address to be assigned to this network interface for the Content Runtime virtual machine. If provided, the address needs to be static.
Portable private netmask		Portable private netmask for the private IP added to the new Content Runtime virtual machine. Required when assigning a portable IP address.
Portable private gateway		Portable private gateway address to be assigned to the new Content Runtime virtual machine. Required when assigning a portable IP address.

Additional Settings

Parameter	Default Value	Description
Network Visibility		The network connection associated with the Content Runtime virtual machine instance. If network visibility is private, connection must exist from the IBM Cloud Pak for AIOps to the network connection associated with the Content Runtime virtual machine being created.

Managed services Parameters

Parameter	Default Value	Description

Parameter	Default Value	Description
Enable FIPS on pattern manager and software repository containers	False	True to enable FIPS on pattern manager and software repository container.
Pattern Manager Access Passphrase		Passphrase used to access the REST API calls to the Pattern Manager.
Pattern Manager Administration Passphrase		Administration passphrase that is used to access the administrative REST API calls to the Pattern Manager.
Key Name for Pattern Manager key set		For Amazon and IBM clouds, the key must exist as a resource. For other provider types, the content is simply the name associated with the key.
Private Key for Pattern Manager - Base 64 encoded		Private key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must be base64 encoded and must be associated with the public key named by the Key Name for Pattern Manager key set.
Public Key for Pattern Manager		Public key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must match the public key named by the Key Name for Pattern Manager key set.
User's Public Key		Public key that is added to the Content Runtime virtual machine's `authorized_keys` file so as to allow access using the associated private key.
Content Runtime Host Name	ibm-content-runtime	The host name of the virtual machine being created to be used as the Content Runtime. The name is just the host name and not the fully qualified host name.

Docker & Chef Settings

Parameter	Default Value	Description
Docker Registry	hub.docker.com	Docker registry to pull the Content Runtime Docker images. The images by default reside on the public Docker hub. The value is read only. It provides you the location where an image is pulled.
Chef License Agreement		Newer Chef Server, Client, Chef DK require license agreement with Chef for commercial. By typing Agree to this field you confirm that you are either using old chef version or you have the required license agreement.
ChefDK version	3.0.36	Stable version of Chef DK that will be installed on pattern manager container.
Docker Enterprise Edition Repository URL		This is an optional parameter. It is mandatory only if you use RHEL as the base image without Docker installed. RHEL supports Docker Enterprise Edition only. After you purchase the license, get the Docker Enterprise Edition Repository URL from https://store.docker.com/my-content. Docker Community Edition is used by default on other supported operating system.
Chef server Organization Name	chef-org	Chef server Organization name that is associated with the configuration of the Chef server.
Chef Administrator User name	chef-admin	Chef administrator user name that is associated with the configuration of the Chef server.
Install Chef Cookbooks	true	Should the Chef cookbooks be installed on the provided Chef server?
Chef Server FQDN		Fully qualified domain name of a remote Chef server. Consists of the server's hostname and domain.
Chef Server IP Address		IPv4 address of a remote Chef server.
Chef Server Certificate - Base 64 encoded		Certificate (.pem file) of the remote Chef server encoded in base64.

Software Repository Settings

Parameter	Default Value	Description
Software Repository User name	repouser	User name to access the Software Repository.
Password for Software Repository		Password for Software Repository User. The combination of user name and password is required to access the Software Repository.
Software Repository Port Number	8888	Port number for Software Repository HTTP access.
Software Repository Secure Port Number	9999	Port number for Software Repository HTTPS access.

Amazon Web Services Content Runtime with existing Chef Server template variables

The following tables list the Amazon template variables.

Cloud Provider

Parameter	Default Value	Description
AWS User ID	ubuntu	An internal field that is needed wherever an ID is required for an image.
Security Group ID		The ID of security group that exists in Amazon cloud. The security group is used while creating the Content Runtime virtual machine.
VPC Subnet ID		The VPC Subnet ID to use while creating Content Runtime. The subnet must exist in Amazon.
Amazon EC2 Instance Type	m3.large	The type of the Amazon EC2 Instance. The default setting is the recommended size for running the Content Runtime.
AWS region		The AWS region to create Content Runtime virtual machine

Cloud Provider Optional Settings

Parameter	Default Value	Description
Private IP Address		Static IPv4 address to be assigned to Content Runtime virtual machine. The provided IP address must be associated to the subnet id. If empty, AWS assigns an IP address that is associated to the subnet id.

Additional Settings

Parameter	Default Value	Description
Network Visibility		The network connection that is associated with the Content Runtime virtual machine instance. If network visibility is private, connection must exist between IBM Cloud Pak for AIOps and the network connection. The network connection is associated with the Content Runtime virtual machine that is being created.

Managed services Parameters

Parameter	Default Value	Description
Enable FIPS on pattern manager and software repository containers	False	True to enable FIPS on pattern manager and software repository container.
Pattern Manager Access Passphrase		Passphrase used to access the REST API calls to the Pattern Manager.
Pattern Manager Administration Passphrase		Administration passphrase that is used to access the administrative REST API calls to the Pattern Manager.
Key Name for Pattern Manager key set		For Amazon and IBM clouds, the key must exist as a resource. For other provider types, the content is simply the name associated with the key.
Private Key for Pattern Manager - Base 64 encoded		Private key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must be base64 encoded and must be associated with the public key named by the Key Name for Pattern Manager key set.
Public Key for Pattern Manager		Public key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must match the public key named by the Key Name for Pattern Manager key set.
User's Public Key		Public key that is added to the Content Runtime virtual machine's `authorized_keys` file so as to allow access using the associated private key.
Content Runtime Host Name	ibm-content-runtime	The host name of the virtual machine being created to be used as the Content Runtime. The name is just the host name and not the fully qualified host name.

Docker & Chef Settings

Parameter	Default Value	Description
Docker Registry	hub.docker.com	Docker registry to pull the Content Runtime Docker images. The images by default reside on the public Docker hub. The value is read only. It provides you the location where an image is pulled.

Parameter	Default Value	Description
Chef License Agreement		Newer Chef Server, Client, Chef DK require license agreement with Chef for commercial. By typing Agree to this field you confirm that you are either using old chef version or you have the required license agreement.
ChefDK version	3.0.36	Stable version of Chef DK that will be installed on pattern manager container.
Docker Enterprise Edition Repository URL		This is an optional parameter. It is mandatory only if you use RHEL as the base image without Docker installed. RHEL supports Docker Enterprise Edition only. After you purchase the license, get the Docker Enterprise Edition Repository URL from https://store.docker.com/my-content. Docker Community Edition is used by default on other supported operating system.
Chef server Organization Name	chef-org	Chef server Organization name that is associated with the configuration of the Chef server.
Chef Administrator User name	chef-admin	Chef administrator user name that is associated with the configuration of the Chef server.
Install Chef Cookbooks	true	Should the Chef cookbooks be installed on the provided Chef server?
Chef Server FQDN		Fully qualified domain name of a remote Chef server. Consists of the server's hostname and domain.
Chef Server IP Address		IPv4 address of a remote Chef server.
Chef Server Certificate - Base 64 encoded		Certificate (.pem file) of the remote Chef server encoded in base64.

Software Repository Settings

Parameter	Default Value	Description
Software Repository User name	repouser	User name to access the Software Repository.
Password for Software Repository		Password for Software Repository User. The combination of user name and password is required to access the Software Repository.
Software Repository Port Number	8888	Port number for Software Repository HTTP access.
Software Repository Secure Port Number	9999	Port number for Software Repository HTTPS access.

VMware Content Runtime with Existing Chef Server template variables

The following tables list the VMware template variables.

Cloud Provider

Parameter	Default Value	Description
vSphere Datacenter		The name of a Datacenter in which to launch the virtual machine.
vSphere Disk Datastore		Datastore for the created disks. Two disk are created, one for the OS template, the other for the data including the software repository content.
vSphere Default Resource Pool		Name of the default resource pool for the cluster. Specified as 'cluster_name/resource_pool'
vSphere Network Adapter Type	vmxnet3	Network interface type
Domain name		Content Runtime Virtual Machine's domain name
CPU cores		The number of CPU cores to allocate for the Content Runtime. The default value is sufficient for the Content Runtime.
RAM		The amount of RAM to allocate for the virtual machine. The default value is sufficient for the Content Runtime.
OS Template Name		The folder/template name to use to configure the OS on the Content Runtime Virtual Machine.
Size of image disk (in GBs)	25	The size of the disk for the Content Runtime. It must be equal or larger than the provided image

Parameter	Default Value	Description
Size of data disk (in GBs)	100	The size of the data disk for the Content Runtime, and Software Repository.
Network interface label		Label to assign to this network interface.
Static IPv4		Static IPv4 address to be assigned to this network interface for the Content Runtime virtual machine. The address must be a static IP.
Gateway IP address		IPv4 gateway address to be used.
IPv4 prefix length (CIDR)	24	Integer value between 1 and 32 for the prefix length, CIDR, to use when statically assigning an IPv4 address
Template image's user name		The user name to use while configuring the Content Runtime virtual machine.
Template image's password		The password to use for the provided SSH user name while configuring the Content Runtime virtual machine. The password is optional if an SSH key is provided.
Template image's private key for SSH connection - Base 64 encoded		A base 64 encoded SSH private key used for configuring the Content Runtime virtual machine. This key is optional if a password is specified.

Cloud Provider Optional Settings

Parameter	Default Value	Description
vSphere Folder		Name of the vSphere folder where the virtual machine is created.
Keep Disks On Remove	false	Should disks be kept on vSphere even if the virtual machine is destroyed?
DNS servers		List of DNS servers for the virtual network adapter. For more than one DNS, the values can be entered by adding more fields.
DNS suffix list		List of DNS suffixes for the virtual network adapter. For more than one suffix, the values can be entered by adding more fields.

Additional Settings

Parameter	Default Value	Description
Network Visibility	public	The network connection associated with the Content Runtime virtual machine instance. If network visibility is private, connection must exist from the IBM Cloud Pak for AIOps to the network connection associated with the Content Runtime virtual machine being created.
Prerequisite checker strictness		If set to strict, the prerequisite checker stops execution if one or more requirements is not installed. If set to lenient, If set to lenient, the prerequisite checker proceeds to install any missing requirement.

Managed services Parameters

Parameter	Default Value	Description
Enable FIPS on pattern manager and software repository containers	False	True to enable FIPS on pattern manager and software repository container.
Pattern Manager Access Passphrase		Passphrase used to access the REST API calls to the Pattern Manager.
Pattern Manager Administration Passphrase		Administration passphrase that is used to access the administrative REST API calls to the Pattern Manager.
Key Name for Pattern Manager key set		For Amazon and IBM clouds, the key must exist as a resource. For other provider types, the content is simply the name associated with the key.
Private Key for Pattern Manager - Base 64 encoded		Private key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must be base64 encoded and must be associated with the public key named by the Key Name for Pattern Manager key set.
Public Key for Pattern Manager		Public key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must match the public key named by the Key Name for Pattern Manager key set.
User's Public Key		Public key that is added to the Content Runtime virtual machine's `authorized_keys` file so as to allow access using the associated private key.
Content Runtime Host Name	ibm-content-runtime	The host name of the virtual machine being created to be used as the Content Runtime. The name is just the host name and not the fully qualified host name.

Docker & Chef Settings

Parameter	Default Value	Description

Parameter	Default Value	Description
Docker Registry	hub.docker.com	Docker registry to pull the Content Runtime Docker images. The images by default reside on the public Docker hub. The value is read only. It provides you the location where an image is pulled.
Chef License Agreement		Newer Chef Server, Client, Chef DK require license agreement with Chef for commercial. By typing Agree to this field you confirm that you are either using old chef version or you have the required license agreement.
ChefDK version	3.0.36	Stable version of Chef DK that will be installed on pattern manager container.
Docker Enterprise Edition Repository URL		This is an optional parameter. It is mandatory only if you use RHEL as the base image without Docker installed. RHEL supports Docker Enterprise Edition only. After you purchase the license, get the Docker Enterprise Edition Repository URL from https://store.docker.com/my-content. Docker Community Edition is used by default on other supported operating system.
Chef server Organization Name	chef-org	Chef server Organization name that is associated with the configuration of the Chef server.
Chef Administrator User name	chef-admin	Chef administrator user name that is associated with the configuration of the Chef server.
Install Chef Cookbooks	true	Should the Chef cookbooks be installed on the provided Chef server?
Chef Server FQDN		Fully qualified domain name of a remote Chef server. Consists of the server's hostname and domain.
Chef Server IP Address		IPv4 address of a remote Chef server.
Chef Server Certificate - Base 64 encoded		Certificate (.pem file) of the remote Chef server encoded in base64.

Software Repository Settings

Parameter	Default Value	Description
Software Repository User name	repouser	User name to access the Software Repository.
Password for Software Repository		Password for Software Repository User. The combination of user name and password is required to access the Software Repository.
Software Repository Port Number	8888	Port number for Software Repository HTTP access.
Software Repository Secure Port Number	9999	Port number for Software Repository HTTPS access.

Other Content Runtime template variables

The following tables list the Other template variables.

Cloud Provider

Table. Cloud provider

Parameter	Default Value	Description
Static IPv4 address		Static IPv4 to assign to this network interface. This must be a static IP, and it is the IP address of the machine as known to the network.
Virtual Machine's Domain Name		Content Runtime Virtual Machine's domain name
Operating System User name		The user name which can be used to sign on to the system to configure the machine as a Content Runtime.
Operating System password		Password for signing onto the system to configure the machine as a Content Runtime. If an SSH key is provided this parameter is not needed.
OS SSH Private Key - Base64 encoded		The SSH key to connect to the virtual machine to allow for configuration of the Content Runtime. The key is optional if the password is provided for the operating system user name. The key is base64 encoded.

Additional Settings

Table. Additional settings

Parameter	Default Value	Description
Offline	False	Is this offline installation.
Network Visibility	public	The network connection associated with the Content Runtime virtual machine instance. If network visibility is private, connection must exist from the IBM Cloud Pak for AIOps to the network connection associated with the Content Runtime virtual machine being created.
Prerequisite checker strictness		If set to strict, the prerequisite checker stops execution if one or more requirements is not installed. If set to lenient, the prerequisite checker proceeds to install any missing requirement.
File system encryption passphrase		Passphrase for encrypted drives.

Managed services Parameters

Table. Managed services parameters

Parameter	Default Value	Description
Enable FIPS on pattern manager and software repository containers	False	True to enable FIPS on pattern manager and software repository container.
Pattern Manager Access Passphrase		Passphrase used to access the REST API calls to the Pattern Manager.
Pattern Manager Administration Passphrase		Administration passphrase that is used to access the administrative REST API calls to the Pattern Manager.
Key Name for Pattern Manager key set		For Amazon and IBM clouds, the key must exist as a resource. For other provider types, the content is simply the name associated with the key.
Private Key for Pattern Manager - Base 64 encoded		Private key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must be base64 encoded and must be associated with the public key named by the Key Name for Pattern Manager key set.
Public Key for Pattern Manager		Public key to be used by Pattern Manager to communicate with the newly instantiated virtual machine. The key must match the public key named by the Key Name for Pattern Manager key set.
User's Public Key		Public key that is added to the Content Runtime virtual machine's `authorized_keys` file so as to allow access using the associated private key.

Docker and Chef Settings

Table. Docker and Chef Settings

Parameter	Default Value	Description
Chef Client Installation Binaries Path (Required only for Offline install)		Provide the local path to chef client installation binaries. The format must be /path/to/chef-clients without without any trailing slash. This value is required if Offline is set to True.
Docker Registry	hub.docker.com	Docker registry to pull the Content Runtime Docker images. The images by default reside on the public Docker hub. The value is read only. It provides you the location where an image is pulled.
Chef License Agreement		Newer Chef Server, Client, Chef DK require license agreement with Chef for commercial. By typing Agree to this field you confirm that you are either using old chef version or you have the required license agreement.
ChefDK version	3.0.36	Stable version of Chef DK that will be installed on pattern manager container.
Docker Enterprise Edition Repository URL		This is an optional parameter. It is mandatory only if you use RHEL as the base image without Docker installed. RHEL supports Docker Enterprise Edition only. After you purchase the license, get the Docker Enterprise Edition Repository URL from https://store.docker.com/my-content. Docker Community Edition is used by default on other supported operating system.
Chef server Organization Name	chef-org	Chef server Organization name that is associated with the configuration of the Chef server.
Chef Administrator User name	chef-admin	Chef administrator user name that is associated with the configuration of the Chef server.
Install Chef Cookbooks	true	Should the Chef cookbooks be installed on the provided Chef server?
Chef Server FQDN		Fully qualified domain name of a remote Chef server. Consists of the server's hostname and domain.

Parameter	Default Value	Description
Chef Server IP Address		IPv4 address of a remote Chef server.
Chef Server Certificate - Base 64 encoded		Certificate (.pem file) of the remote Chef server encoded in base64.

Optional Settings for Offline Installation

Table. Optional Settings for Offline Installation

Parameter	Default Value	Description
Docker installation file		Optional file location of a docker .deb/.rpm file to be installed on the system. If the installation package is stored within the content runtime system, for example, docker.deb saved to /tmp, then the input is file:///tmp/docker.deb. If the installation package is stored in a server with HTTP access, then the input is `http://<server_IP>/path/to/docker.deb`.
Docker Compose installation file		Optional file location of a Docker Compose executable file to be installed on the system. If the installation package is stored within the content runtime system, for example, docker-compose saved to /tmp, then the input is `file:///tmp/docker-compose`. If the installation package is stored in a server with HTTP access, then the input is `http://<server_IP>/path/to/docker-compose`.
Software Repository Docker image location		Optional file location of the Software Repository Docker image to be loaded to the system. If the installation package has been stored within the content runtime system, for example, camc-sw-repo saved to /tmp, then the input is `file:///tmp/camc-sw-repo`. If the installation package is stored in a server with HTTP access, then the input is `http://<server_IP>/path/to/camc-sw-repo`.
Pattern Manager Docker image location		Optional file location of the Pattern Manager Docker image to be loaded to the system. If the installation package is stored within the content runtime system, for example, camc-pattern-manager saved to /tmp, then the input is `file:///tmp/camc-pattern-manager`. If the installation package is stored in a server with HTTP access, then the input is `http://<server_IP>/path/to/camc-pattern-manager`.

Software Repository Settings

Table. Software Repository Settings

Parameter	Default Value	Description
Software Repository User name	repouser	User name to access the Software Repository.
Password for Software Repository		Password for Software Repository User. The combination of user name and password is required to access the Software Repository.
Software Repository Port Number	8888	Port number for Software Repository HTTP access.
Software Repository Secure Port Number	9999	Port number for Software Repository HTTPS access.

Provisioning an offline Content Runtime

It is possible to deploy a Content Runtime on an existing virtual machine with limited or no internet connectivity.

To provision an offline Content Runtime, a virtual machine must exist with a static IP address and the installed operating system in the virtual machine must be one of the supported Linux distributions. In addition, either manually download software prerequisites by using the steps in Download prerequisite software section or install the software prerequisites on the virtual machine before the deployment as described in Pre-install prerequisite software manually.

General steps

- Deploy a system to host the Content Runtime
- Download prerequisite software
- Copy software to the Content Runtime host system
- Load Content Runtime Templates
- Deploy an offline Content Runtime

- Optional - Pre-install prerequisite software manually (follow this step if you do not want to use the helper script that is described in Download prerequisite software to download the prerequisite software.)

Deploy a system to host the Content Runtime

For an offline deployment of the Content Runtime, a virtual machine must meet the general infrastructure requirements and the offline template installation requirements.

After your content runtime virtual machine is setup, follow the instructions in Download prerequisite software to download the software prerequisite and make them available during automation, or proceed to Pre-install prerequisite software manually to install the software prerequisite yourself before the offline content runtime deployment.

Download prerequisite software

When you deploy an offline content runtime as described in Deploy an offline Content Runtime, the automation installs all the software prerequisites. But you are required to download the binary and provide the location of the binary during deployment. A set of helper scripts is available in a public GitHub repository to assist you in gathering the supported versions of the required software prerequisites. This helper script downloads the following required software:

- Docker
- Docker Compose
- Chef server
- Chef client
- Chef DK
- CAMC Pattern Manager Docker Image
- CAMC Software Repository Docker Image

Follow the steps to execute the helper script that downloads the required software prerequisites.

- Clone helper repository

```
bash
git clone https://github.com/IBM-CAMHub-Open/IBM-CAMHub-Open.git
```

- Execute prereq-downloader.sh

The **prereq-downloader.sh** script that is cloned in the previous step can be used to download all software prerequisites for an offline Content Runtime deployment.

```
bash
cd IBM-CAMHub-Open/bin/prereqDownloader
./prereq-downloader.sh <platform> <platform_version> <optional:release>
```

Where **<platform>** and **<platform_version>** are the Linux distribution and version of the virtual machine to which the Content Runtime is being deployed. The optional parameter **<release>** provides a way to obtain the required software for the older versions based on the environment's needs.

For example, downloading requirements for a Content Runtime installation on a virtual machine running Ubuntu 18.04

```
./prereq-downloader.sh ubuntu 18.04 4.0
```

Requirements get downloaded to a folder named **prereqs_<platform>_<platform_version>_<release>** which can later be copied to an HTTP server or the target virtual machine.

Known issues and workarounds

- If the script execution fails with the following error, then you may have encountered a GitHub API rate hit limit:

  "No repositories found, possibly the connection to the host https://github.com/IBM-CAMHub-Open failed"

  To overcome the rate limit, set the following environment variable and rerun the script:

  ```
  export CloneWithToken=YOUR_GITHUB_TOKEN@
  ```

  Note: You must append @ to your GitHub token.

- If **prereq-downloader.sh** execution fails due to curl auth error, then add **--insecure** parameter at the end of the curl.

Copy software to the Content Runtime host system

Copy software from the **prereqs_<platform>_<platform_version>_<release>** directory created in the previous step to the system which hosts the Content Runtime.

The file directory where the software is copied is used as input during Content Runtime deployment.

The deployment process requires the **IBM-CAMHub-Open.tar** produced in the previous step to be copied to **/var/** directory on the system that hosts the Content Runtime.

Alternatively, files can be placed on an accessible HTTP Server. In this case, the HTTP location is used as input to the Content Runtime deployment.

Running on a Chef 12 environment

If a Content Runtime is meant to be deployed using Chef client version 12, a version which is no longer officially supported, a third parameter can be specified when downloading requirements. If the **release** parameter is set to **1.0**, Chef client version 12 is downloaded along with other compatible versions of the requirements.

Load Content Runtime Templates

The Content Runtime templates need to be manually loaded into a Managed services environment that does not have access to github.com.

The recommended path is to first import the Content Runtime templates to a local GitLab or GitHub Enterprise. Content Runtime templates are available in Git repositories downloaded in the previous step and saved in **IBM-CAMHub-Open_advanced_content_runtime.tar**.

After the templates are imported to GitLab or GitHub Enterprise, the desired Content Runtime templates can be loaded into Managed services by creating a template in the user interface.

If GitLab or GitHub Enterprise are not available, the script **loadContentRuntimeTemplates.sh** in **IBM-CAMHub-Open_advanced_content_runtime.tar** can be executed to load the Content Runtime templates into Managed services.

After the Content Runtime templates have been loaded into Managed services, confirm whether they appear in the dropdown under **Manage** > Content runtimes > Create content runtime.

Deploy an offline Content Runtime

Before you deploy your offline content runtime, make sure your content runtime virtual machine meets all the requirement specified in general infrastructure requirements page and offline template installation requirements section.

1. Select the **Other** Cloud Provider after you click **Create content runtime**. Name the instance and click **Create**.
2. Provide the information required for establishing a connection with the existing virtual machine, the Docker and Chef settings, and other parameters.

   Note: The value for **Chef Client Installation Binaries Path** must be provided. The value must be a local path to the binary without trailing slash. If your chef client binary is in directory /tmp/chef-clients then the value is as follows:

   ```
   /tmp/chef-clients
   ```

3. In the section **7. Optional Settings for Offline Installation**, five optional fields are displayed: Docker installation file, Docker Compose installation file, Chef Server installation file, Pattern Manager Docker image location and Software Repository Docker image location. These inputs must be filled out by either providing the file path to the installation binaries or the HTTP location from where they can be downloaded.

   - **File location**: If the installation package has been stored in a location within the system, for example, **docker.deb** saved to **/tmp**, the correct input would be **file:///tmp/docker.deb**. The full path, including file name, has to be provided.
   - **HTTP location**: Similar to the file location, if the installation package was stored in a server with enabled HTTP access, the correct input would be **http://<server_IP>/path/to/docker.deb**.

Optional - Pre-install prerequisite software manually

Instead of using the helper script described in Download prerequisite software, you may choose to download the prerequisite software manually and provide the location of the binary during content runtime deployment. Alternatively, you may choose to download the prerequisite software manually, install them and during content runtime deployment, leave the section **Optional Settings for Offline Installation** empty. The following section describes the software you need to download in both the cases. After you download or download and installed the listed software, you can follow the steps in Deploy an offline Content Runtime to deploy the offline content runtime.

Before you proceed, make sure your content runtime virtual machine meets all the requirements specified in general infrastructure requirements page and offline template installation requirements section.

Installation binaries

An internet connection is required to obtain the required software packages. These installation binaries need to either be installed previously on the virtual machine or stored in locations that can be accessed during the deployment. These files are:

- Docker
- Docker Compose
- Chef server

These installation binaries need to be downloaded and stored in a location on content runtime virtual machine that can be accessed during the deployment

- Chef client
- Chef DK

Docker, Chef Server, Chef DK and Chef Client need to be obtained in a **.deb** (Ubuntu) or **.rpm** (RHEL/CentOS) format depending on the Linux distribution to which the Content Runtime is deployed. Content runtime deploy has been validated with Chef Server 12.17.33 and Chef Client 14.0.190.

In addition to the programs mentioned previously, two Docker images from DockerHub need to be manually downloaded from a machine with Docker installed:

- CAMC Pattern Manager
- CAMC Software Repository

Obtaining Docker images

The two docker images can be downloaded from the public DockerHub repository by running the following commands:

```
bash
$ docker pull ibmcom/camc-pattern-manager:3.0-current
$ docker pull ibmcom/camc-sw-repo:2.0-current
```

This downloads and installs the Docker images in the current machine, to obtain a file that can be shared with another machine, the **docker save** command can be used as follows

```
bash
$ docker save ibmcom/camc-sw-repo:3.0-current > camc-sw-repo
$ docker save ibmcom/camc-pattern-manager:2.0-current > camc-pattern-manager
```

This creates two files **camc-sw-repo** and **camc-pattern-manager** that contains the images.

Offline template deployment installation requirements

The deployment process can handle the installation of the provided **.deb** or **.rpm** files. However, these packages may have requirements that need to be manually installed before starting the process. These packages can be installed through **apt-get install** or **yum install** commands.

Requirements can be obtained from each **.deb** installation package by running the following commands:

```
bash
# Ubuntu
dpkg -I <package>.deb

# Red Hat and CentOS
yum deplist <package>.rpm
```

Docker installation

Ubuntu:

```
    iptables
    init-system-helpers
    lsb-base
    libapparmor1
    libc6
    libdevmapper
    libltdl7
    libseccomp2
    libsystemd0
```

Red Hat and CentOS:

```
    device-mapper-devel
    glibc-static
    libseccomp
    libseccomp-devel
```

```
    libselinux-devel
    libtool-ltdl-devel
    selinux-policy
    selinux-policy-devel
    systemd-devel
```

Note: You can obtain the selinux related packages if you download container-selinux rpm.

Local Knife Configuration

Configuring the Knife command to work with CAM.

After the Content Runtime is deployed, a Chef Server is available with an initial set of Chef Recipes and Roles deployed. To manage Chef Content, install the Chef DK on a local Workstation. The Chef DK installs the Knife tool which is the primary tool for managing the installed Chef Server.

Before you begin

Prior to beginning, identify the following information from the Content Runtime deployment.

Parameter	Description
User	User to connect to the Chef Server
Password	Password to connect to the Chef Server
Chef Hostname	Full Host name of the Chef Server, property=chef_fqdn
Chef Org	Default Chef Organization, property=chef_org
Chef Admin Name	Chef administration user, property=chef_admin

Ensure the fully qualified Chef Server is resolvable from the local workstation.

Installing the Chef DK

The Knife Command is packaged with the Chef Development Kit (Chef DK) which must first be installed on a local workstation. The Chef DK package will make the knife command available.

To begin, follow the instructions published by opscode on this topic here -> Install the Chef DK.

Configuring Knife

Understanding Knife configuration

Review the following documentation on the Chef DK and knife command:

- Workstation Configuration
- Knife Documentation

Creating the chef-admin.pem file

Knife requires authentication for the Chef Admin user to properly manage the Chef Server. Knife authenticates with a .pem file which must be copied from the chef server.

1. From the Chef Server, locate the /etc/opscode/{Chef Admin}.pem file.
2. Copy to your local workstation.

Minimal knife.rb

1. Create a directory to hold the knife configuration, you may need one per Content Runtime.
2. Copy the {Chef Admin}.pem file to this directory (such as `chef-admin.pem`).
3. Create a file called knife.rb with the following minimal values.

```
current_dir = File.dirname(__FILE__)
log_level                :info
log_location             STDOUT
node_name                '{Chef Admin}'
client_key               "{Knife Dir}/{Chef Admin}.pem"
chef_server_url          "https://{Chef Hostname}/organizations/{Chef Orgname}"
```

Testing Knife

Knife commands will by default require the certificate of the server present in the trusted_cert directory. In order to force this, run the following command.

```
knife ssl fetch
```

You can now test simple knife functions.

- Listing cookbooks:

  ```
  knife cookbook list
  ```

  Output:

  ```
  db2                 1.0.4
  httpd               1.4.0
  ibm_cloud_utils     1.0.4
  ihs                 1.0.5
  im                  1.0.3
  linux               1.0.4
  oracle_mysql        1.0.4
  oracledb            1.0.4
  tomcat              1.0.3
  was                 1.0.6
  wasliberty          1.0.5
  wmq                 1.0.4
  ```

- Listing Chef Roles:

  ```
  knife role list
  ```

  Output:

  ```
  db2_create_db
  db2_v105_install
  db2_v111_install
  httpd24-base-install
  httpd24-ssl-vhosts
  ihs-liberty-admin
  ihs-liberty-nonadmin
  ihs-wasmode-admin
  ihs-wasmode-nonadmin
  liberty_core_install
  ```

- Viewing deployed nodes:

  ```
  knife node list
  ```

  Output:

  ```
  nschambu-mq-01
  nschambu-mq-02
  nschambu-was-01
  nschambu-was-02
  ```

For more knife commands refer to the Knife Documentation

The Pattern Manager has exposed an API to automate the management of Chef Cookbooks. For more information, see Pattern Manager API.

Modifying a deployed content runtime

Modifying a content runtime after it has been deployed.

You can redeploy an existing content runtime instance after selecting the desired template version and modifying the parameters. Full instructions for modifying a deployed instance can be found at modifying a deployed template instance.

The taint request described on that page might not be necessary for modifying your content runtime instance. Not all properties are eligible for modification. Some properties might be unavailable while others might not trigger a resource update during the plan request.

Modifying an existing deployment can take two paths: change or destroy and re-create. A change request updates the property in Terraform and makes a request to the affected resources in the content runtime. A destroy and re-create request destroys your content runtime virtual machine and create a new one with the updated properties.

Modify scenarios

Most of property updates require the content runtime virtual machine to be destroyed and re-created, which is destructive. For this reason, you are recommended to use modify in the following scenarios.

Redeploying a failed or destroyed instance

Template modification can be used for more than just modifying deployed content runtime instances. It can also be used to redeploy a failed or destroyed content runtime. Update the parameters that caused the deployment to fail and try the deployment again without having to reenter all of the parameters. Similarly, if a content runtime is accidentally destroyed, it can be redeployed with all of the same parameters from the destroyed copy.

Note: Redeploying a destroyed instance does not re-create the Chef server contents and you must recover the Content Runtime.

Updating the pattern manager key pair or software repository password

The content runtime allows modification of some Managed services properties in addition to the hardware-specific properties. These include the software repository password and the pattern manager key pair properties, which are the pattern manager private key, pattern manager public key, and the key name for pattern manager key set. Updating these properties runs an update script

Troubleshooting destroy and add in the plan output

It is important that you pay attention to the output of the plan request as it includes whether the existing content runtime virtual machine gets destroyed as part of the modify process. Destroying the content runtime virtual machine results in the creation of a new Chef server that is unaware of any node virtual machines that have been previously deployed to this content runtime instance. If you have mistakenly destroyed your content runtime virtual machine as part of a modify process and want to reregister your existing template instances, see Recovering the Content Runtime.

If the destroy and add is related to a resource that contains singlenode or single-node, then your modify action destroys your content runtime VM. It is not recommended that you continue this action unless you are redeploying a failed instance or if your content runtime has no nodes assigned to it. Here is a sample plan output that destroys the content runtime virtual machine in an IBM Cloud content runtime:

```
The following execution plan has been generated.
Resource actions are indicated with the following symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:
-/+ ibm_compute_vm_instance.single-node (new resource required)
id: "46145801" => (forces new resource)
block_storage_ids.#: "0" =>
cores: "4" => "4"
datacenter: "dal09" => "dal09"
dedicated_acct_host_only: "true" => "true"
disks.#: "2" => "2"
disks.0: "25" => "25"
disks.1: "100" => "100"
domain: "stglabs.ibm.com" => "stglabs.ibm.com"
file_storage_ids.#: "0" =>
hostname: "ibm-content-runtime" => "ibm-content-runtime"
hourly_billing: "true" => "true"
ip_address_id: "76112907" =>
ip_address_id_private: "74680447" =>
ipv4_address: "169.53.243.171" =>
ipv4_address_private: "10.98.30.204" =>
ipv6_address: "" =>
ipv6_address_id: "" =>
ipv6_enabled: "false" => "false"
local_disk: "false" => "false"
memory: "8192" => "8192"
network_speed: "1000" => "1000"
os_reference_code: "UBUNTU_14_64" => "UBUNTU_14_64"
private_network_only: "false" => "false"
private_subnet: "10.98.30.192/26" =>
private_vlan_id: "1601131" => "-1" (forces new resource)
public_bandwidth_limited: "" =>
public_bandwidth_unlimited: "false" => "false"
public_ipv6_subnet: "" =>
public_vlan_id: "1601127" =>
secondary_ip_addresses.#: "0" =>
ssh_key_ids.#: "1" => "1"
ssh_key_ids.1040221: "1040221" => "1040221"
wait_time_minutes: "90" => "90"

-/+ null_resource.call_launcher (new resource required)
id: "2023496446251967403" => (forces new resource)
triggers.%: "4" => (forces new resource)
triggers.cr_instance_ids: "46145801" => "" (forces new resource)
triggers.private_key_changed: "******" => "" (forces new resource)
triggers.public_key_changed: "ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAADAQABAAABAQCunj1Un1T9s1FNdHl+yY9ta5b71ldQahM7CKYnSKf+Y8baXmEejajSM+cc2WXhcXZPscjate8P7oLdQ
fFg/zAr1MzbKU7sTvupA6Qx+hlneweL0KKhGpMFLhsp9lgFzGLW4424k+WvKlUf3o59SbLwaMIVeMM/0DlOGLfh2Z2e7o39fsLIg6D5ptr8hB
nsQZ9zwReh9qbwiAW9bbl/5tYgit0HSucrYiZRTss8unHCeuVvRwxK6KeVoqzNRxx2UtL0eb62TzVM+IIm4JlsXr1Nx6IvLf/z6icaMViu3GP
Ktf4Gz5zbZv3PI/EF/7CJf03S5F5vYuDCi4CKa0OhKh9d" => "" (forces new resource)
triggers.repo_pass_changed: "repopass" => "" (forces new resource)


Plan: 2 to add, 0 to change, 2 to destroy.
------------------------------------------------------------------------
This plan was saved to: terraform_execution_plan
To perform exactly these actions, run the following command to apply:
terraform apply "terraform_execution_plan"
```

In the preceding output, there are two resources being destroyed and then redeployed: ibm_compute_vm_instance.single-node and null_resource.call_launcher. As stated before, destroying ibm_compute_vm_instance.single-node destroys your content runtime VM. It is not recommended that you apply this plan.

IBM Cloud and the private VLAN ID

Looking more closely at the output from the previous section, you can see that the property change that is forcing a new resource is the private_vlan_id being set from 1601131 to -1. The private VLAN name was not set during the initial deploy so the IBM Cloud assigned our instance a default private VLAN based on the datacenter. If this is the only reason why the content runtime virtual machine is set to be destroyed, then find out the VLAN name associated with that ID in the IBM Cloud to set that parameter for our content runtime deploy.

Using the ID, log into the IBM Cloud to find the VLAN name associated with our deploy. Find and select your virtual machine in the Device List to open the Device Details page. There will be a network section on this page, which includes the private network information. Find the private VLAN assigned to your device and click it to open the VLAN Details page. The VLAN Details page contains the name of your private VLAN. Copy that name and go back to the Managed services UI.

Go back to your content runtime instance and click the modify tab. Find the Private VLAN name field for your content runtime and set it to the VLAN name that was found in the IBM cloud. Update any other parameters that you had previously updated and click plan. The resulting plan should no longer include destroying the ibm_compute_vm_instance.single-node instance.

Updating the Content Runtime Docker Containers

Updating the pattern manager and software repository Docker containers after is has been deployed.

The Content Runtime consists of a virtual machine with two Docker containers - one for the pattern manager and one for the software repository. There are times where you may want to update these Docker containers manually rather than redeploying a new content runtime. The Content Runtime provides a helper script to assist in updating the Docker containers so that you don't need to manually run Docker commands.

Image upgrade script

The image upgrade script is called **image-update.sh** and is found in the **~/advanced-content-runtime** directory on the Content Runtime. Usage syntax is:

```
~/advanced-content-runtime/image-upgrade.sh <container> <version>
```

The container and version variables can be retrieved from the image column of the output of the **docker ps** command. For example, in the following output, the container is **camc-pattern-manager** and the version is 1.0-current.

```
$ docker ps
CONTAINER ID        IMAGE                                      COMMAND                  CREATED
STATUS              PORTS                                          NAMES
a14ae0bd4ae6        ibmcom/camc-pattern-manager:1.0-current    "bin/bash /opt/ibm/p…"   6 hours ago         Up
6 hours           0.0.0.0:5443->443/tcp                          camc-pattern-manager
f0e7aa1c3c41        ibmcom/camc-sw-repo:1.0-current            "/bin/bash /tmp/inst…"   6 hours ago         Up
6 hours           0.0.0.0:8888->8888/tcp, 0.0.0.0:9999->9999/tcp   camc-sw-repo
```

Using these values in the image-upgrade command results in the following request:

```
~/advanced-content-runtime/image-upgrade.sh camc-pattern-manager 1.0-current
```

This will pull the latest 1.0-current docker container and a new image will appear in the output of **docker images** for the specified container.

Deploying a GDPR enabled Content Runtime

As part of GDPR enablement, it is recommended that you encrypt specific directories that may contain personal information. These include directories that contain Content Runtime configuration information and Content Runtime Pattern Manager logs.

Provisioning a GDPR enabled Content Runtime requires using an existing virtual machine with mounted encrypted block devices.

General steps

- Deploy a system to host the Content Runtime
- Encrypt volumes using dm-crypt
- Deploy Content Runtime

Additional information

- Pattern Manager and Software Repository Logs

Deploy a system to host the Content Runtime

For a GDPR enabled deployment of the Content Runtime, a virtual machine meeting infrastructure requirements must exist and be accessible by Managed services. In addition to the general infrastructure requirements, two unformatted disks must be attached to the virtual machine.

The following set of steps contains example commands for configuring a virtual machine where a Content Runtime instance will be deployed. In this example, two logical volumes are created and mounted in well-known locations used by the Pattern Manager.

Encrypt and mount volumes

dm-crypt provides transparent encryption of block devices. You can access the data immediately after you mount the device. For more information about dm-crypt, see https://wiki.archlinux.org/index.php/dm-crypt.

Prerequisites

Ensure that the following packages are installed on all the nodes of your Content Runtime VM:

- On Red Hat Enterprise Linux (RHEL), the following packages must be installed:
  - cryptsetup
  - device-mapper
  - util-linux

    If the packages are not installed, run these commands as a root user to install them:

    ```
    yum install cryptsetup-luks util-linux
    ```
- On Ubuntu, the following packages must be installed:
  - cryptsetup
  - libdevmapper1
  - util-linux
  - lvm2
  - xfsprogs

    If the packages are not installed, run these commands as a root user to install them:

    ```
    apt-get install cryptsetup util-linux lvm2 xfsprogs
    ```

Create a partition table

Before encrypting volumes, a label and at least one partition must be created in the drive.

1. Use the **parted** tool on the disk

   ```
   parted <full path and name of the physical volume>
   ```

   Example commands and output

```
parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

2. Add a partition table to the volume using `mklabel`

   Example commands

   ```
   (parted) mklabel gpt
   ```

3. Create a partition, in this case a single partition will be created using all available space. **Note:** Multiple logical volumes can later be defined inside a single partition.

   Example commands

   ```
   (parted) mkpart primary 1 -1
   ```

4. Exit parted by typing `quit`

5. Repeat steps 1-4 for other disks. In this example, repeat for `/dev/sdc`

Encrypting the volumes

These commands will create and encrypt a logical volume. That logical volumes will be mounted to the corresponding directories the may contain personal information.

Run these commands as root.

1. Configure logical volume management (LVM) to store the encrypted data:

   1. Create a physical volume.

      ```
      pvcreate <full path and name of the partition>
      ```

      Example commands and outputs:

      ```
      pvcreate /dev/sdb1
      Physical volume "/dev/sdb1" successfully created.

      pvcreate /dev/sdc1
      Physical volume "/dev/sdc1" successfully created.
      ```

   2. Create a volume group.

      ```
      vgcreate <name of the volume group> <full path to the physical volume>
      ```

      Example commands and outputs:

      ```
      vgcreate pmvg /dev/sdb1
      Volume group "pmvg" successfully created.

      vgcreate pmlogvg /dev/sdc1
      Volume group "pmlogvg" successfully created.
      ```

   3. Create a logical volume.

      ```
      lvcreate -L <amount of space required> <name of the volume group> -n <name of the logical volume>
      ```

      Example commands and outputs:

      ```
      lvcreate -L1G pmvg -n pmlv
      Logical volume "pmlv" created.

      lvcreate -L5G pmlogvg -n pmloglv
      Logical volume "pmloglv" created.
      ```

2. Create a dm-crypt LUKS Container in the volume.

   ```
   cryptsetup -y luksFormat <full path to the logical volume>
   ```

   If you want to use a passphrase for decrypting, you can specify it now. This passphrase will be needed for deploying the Content Runtime later. The passphrase needs to be common for both logical volumes.

   The following is an example command and output:

   ```
   cryptsetup -y luksFormat /dev/pmvg/pmlv

   WARNING!
   ```

```
========
This will overwrite data on /dev/pmvg/pmlv irrevocably.

Are you sure? (Type uppercase yes): YES
Enter passphrase:
Verify passphrase:

cryptsetup -y luksFormat /dev/pmlogvg/pmloglv

WARNING!
========
This will overwrite data on /dev/pmlogvg/pmloglv irrevocably.

Are you sure? (Type uppercase yes): YES
Enter passphrase:
Verify passphrase:
```

3. (Optional) Add LUKS key to your encrypted device.

   cryptsetup supports adding a key to your logical volume using a keyfile. This keyfile can be added to /etc/crypttab to allow your encrypted drives to be auto-unlocked at boot time. Your keyfile can be random data or something specific. It is recommended that the permissions on your keyfile be set to root only.

   Create keyfile examples:

   ```
   dd if=/dev/random bs=32 count=1 of=/etc/keys/pm_key
   printf "This is a plaintext passphrase key file" > /etc/keys/pmlog_key
   ```

   Add the keyfile to your LUKS partition:

   ```
   cryptsetup luksAddKey <full path to logical volume> <full path to key file>
   ```

   The following is an example command and output:

   ```
   cryptsetup luksAddKey /dev/pmvg/pmlv /etc/keys/pm_key
   Enter any passphrase:

   cryptsetup luksAddKey /dev/pmlogvg/pmloglv /etc/keys/pmlog_key
   Enter any passphrase:
   ```

4. Open the LUKS container and map the logical volume to the volume group.

   ```
   cryptsetup  luksOpen <full path to the logical volume> <name of the logical volume>
   ```

   The following is an example command and output:

   ```
   cryptsetup luksOpen /dev/pmvg/pmlv pmlv
   Enter passphrase for /dev/pmvg/pmlv:

   cryptsetup luksOpen /dev/pmlogvg/pmloglv pmloglv
   Enter passphrase for /dev/pmlogvg/pmloglv:
   ```

5. Create a file system on the logical volume. You can use any file system. The command here is for using an XFS file system.

   ```
   $ mkfs.xfs /dev/mapper/<name of the logical volume>
   ```

   The following is an example command and output:

   ```
   $ mkfs.xfs /dev/mapper/pmlv

   meta-data=/dev/mapper/pmlv       isize=512    agcount=4, agsize=65408 blks
            =                       sectsz=512   attr=2, projid32bit=1
            =                       crc=1        finobt=0, sparse=0
   data     =                       bsize=4096   blocks=261632, imaxpct=25
            =                       sunit=0      swidth=0 blks
   naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
   log      =internal log           bsize=4096   blocks=855, version=2
            =                       sectsz=512   sunit=0 blks, lazy-count=1
   realtime =none                   extsz=4096   blocks=0, rtextents=0

   $ mkfs.xfs /dev/mapper/pmloglv

   meta-data=/dev/mapper/pmloglv     isize=512    agcount=4, agsize=327552 blks
            =                        sectsz=512   attr=2, projid32bit=1
            =                        crc=1        finobt=0, sparse=0
   data     =                        bsize=4096   blocks=1310208, imaxpct=25
            =                        sunit=0      swidth=0 blks
   naming   =version 2               bsize=4096   ascii-ci=0 ftype=1
   log      =internal log            bsize=4096   blocks=2560, version=2
            =                        sectsz=512   sunit=0 blks, lazy-count=1
   realtime =none                   extsz=4096   blocks=0, rtextents=0
   ```

6. Create a mount location to mount the file systems for the pattern manager config directory and docker container log directories

   ```
   mkdir <mount location>
   ```

   The following is an example command

   ```
   mkdir -p /opt/ibm/docker/pattern-manager
   mkdir -p /var/log/ibm/docker
   ```

7. Mount the pattern manager config and docker container log filesystems

   ```
   mount /dev/mapper/<name of the logical volume> <mount location>
   ```

   The following is an example command:

   ```
   mount /dev/mapper/pmlv /opt/ibm/docker/pattern-manager
   mount /dev/mapper/pmloglv /var/log/ibm/docker
   ```

   To automatically mount between system restarts, add the following lines to the /etc/crypttab and /etc/fstab files:

   - Add the following line to /etc/crypttab file, where is a unique name of your choosing:

     ```
     <target name> <full path to the logical volume> {none|<absolute_path_to_keyfile>} luks
     ```

     Where, add **none** if you used a passphrase for decrypting, or add the full path to the key file, if you used a key file for decrypting.

     The following are example lines in /etc/crypttab:

     ```
     pm_crypt /dev/pmvg/pmlv /etc/keys/pm_key luks
     pmlog_crypt /dev/pmlogvg/pmloglv /etc/keys/pmlog_key luks
     ```

   - Add the following line to the /etc/fstab file, where matches the unique name added to /etc/crypttab:

     ```
     /dev/mapper/<target name> <full path to the volume group> xfs defaults 0 2
     ```

     The following are example lines in /etc/fstab:

     ```
     /dev/mapper/pm_crypt /opt/ibm/docker/pattern-manager xfs defaults 0 2
     /dev/mapper/pmlog_crypt /var/log/ibm/docker xfs defaults 0 2
     ```

8. Reboot the virtual machine

Deploy Content Runtime

1. In the CAM UI, click **Manage** > **Content runtimes**.
2. Select the **Other** Cloud Provider after clicking the **Create content runtime** button. Choose any of the three Runtime options as they will all support encrypted volumes. Name the instance and press the **Create** button.
3. Provide the information required for establishing a connection with the existing virtual machine and other parameters. Include the encryption passphrase for the encrypted volumes in the **File system encryption passphrase** field.

Pattern Manager and Software Repository Logs

The Content Runtime's Pattern Manager and Software Repository logs can be found on the content runtime virtual machine under the **/var/log/ibm/docker** directory in the **pattern-manager** and **software-repository** directories, respectively. Should it be required to delete personal data from the Content Runtime logs, this is where you will find them.

Broadcast messages

The following message may be broadcast on your system during the content runtime installation process. It can be ignored.

```
Broadcast message from root@ibm-cam-runtime (Mon 2018-05-14 15:46:50 UTC):

Password entry required for 'Please enter passphrase for disk pmlogvg-pmloglv (pmlogvg)!' (PID 32283).
Please enter password with the systemd-tty-ask-password-agent tool!
```

Troubleshooting and known issues

Troubleshoot your Content Runtime infrastructure and read the known issues.

Failed to create the Content Runtime

The console log displays the following error message which is caused by the issue. You must correct the error and configure the Content Runtime again.

```
null_resource.singlenode: Still creating... (3m40s elapsed)
null_resource.singlenode: Still creating... (3m40s elapsed)
null_resource.singlenode: Still creating... (3m40s elapsed)
null_resource.singlenode: Still creating... (3m40s elapsed)
null_resource.singlenode: Still creating... (3m40s elapsed)
null_resource.singlenode: Still creating... (3m50s elapsed)
null_resource.singlenode: Still creating... (4m0s elapsed)
null_resource.singlenode: Still creating... (4m10s elapsed)
null_resource.singlenode: Still creating... (4m20s elapsed)
null_resource.singlenode: Still creating... (4m30s elapsed)
null_resource.singlenode: Still creating... (4m40s elapsed)
null_resource.singlenode: Still creating... (4m50s elapsed)
null_resource.singlenode: Still creating... (5m0s elapsed)
Error applying plan:

1 error(s) occurred:

* null_resource.singlenode: 1 error(s) occurred:

* ssh: handshake failed: ssh: unable to authenticate, attempted methods [none password], no supported methods
remain

Terraform does not automatically rollback in the face of errors. Instead, your Terraform state file has been
partially updated with any resources that successfully completed. Please address the error above and apply
again to incrementally change your infrastructure.
```

If you are unable to determine the cause of the failure from the logs, it is possible to log in to the Content Runtime virtual machine and run the configuration process manually using the following command which is located in the home directory of the OS user:

```
advanced-content-runtime/launch-docker-compose.sh
```

Additional debug information can be captured from the `./advanced-content-runtime/launch-docker-compose.sh` script by editing the `./advanced-content-runtime/.advanced-runtime-config/.launch-docker-compose.sh` file and adding the line `--debug` to the end of the file. Rerun the `./advanced-content-runtime/launch-docker-compose.sh` command to capture the additional information. Parameters may be updated in the `./advanced-content-runtime/.advanced-runtime-config/.launch-docker-compose.sh` file if the parameter values were the cause of the failure.

If you are successful when completing the creation of a Content Runtime from the command line, you need to complete the following steps to add it to the Content Runtime creation panel:

1. Go back to the Content Runtime creation panel and recreate the Content Runtime from the beginning.
2. Leave the virtual machine intact, go back to the Content Runtime panel, select create Content Runtime's `Other` template. You must enter all the parameters as they appear in the `~/advanced-content-runtime/.advanced-runtime-config/.launch-docker-compose.sh` file in the create panel so that the existing virtual machine is reused. Managed services then stores its information so that it can be used later in pattern deployment.

Common issues when deploying a Content Runtime

The following are a set of input issues that would result in a failed deployment of a Content Runtime, how to identify them by reading the provided `Log File` and the steps to solve them.

All cloud providers

The following errors can occur independently of the selected cloud provider.

```
ssh: unable to authenticate, attempted methods [none password], no supported methods remain
```

- This error is displayed when the credentials (user name and password or private key) provided for the SSH connection to the virtual machine were incorrect.

  ```
  [ERROR] Platform <distribution> not supported
  ```

- If a Content Runtime is trying to be deployed to an unsupported operating system, this error is displayed.

  ```
  [ERROR] This OS version (<version>) is not supported
  ```

- The error occurs when a Content Runtime is trying to be deployed to a supported operating system but an unsupported version.

  ```
  [ERROR] This script requires root permissions with the NOPASSWD option enabled for executing
  ```

- This error is displayed when the user account provided for connecting to the Content Runtime has no **sudo** permissions or creates a prompt for password when executing as **sudo**:
- To remove the password prompt, the file **/etc/sudoers** must be changed to include **user_name ALL=(ALL:ALL) NOPASSWD: ALL**

```
[ERROR] This script requires <package manager> permissions for executing
```

- This error occurs when the user account provided for connecting to the Content Runtime has no permissions to use the OS's package manager. A different user account with higher permissions can be provided to solve it.

```
[ERROR] This script requires at least 5GB of available disk space
```

- This error is displayed when the hard drive being used for installing the necessary software and downloading additional packages does not contain at least 5GB of empty space. An image containing a larger hard drive or freeing up more space should solve this issue.

```
[ERROR] The provided encoded private key contains a password. Pattern Manager requires the use of a
passwordless private key.
```

- This error is displayed when the provided private key was password-encrypted when generated. In order to solve this issue, generate another private key without providing a password.

```
[ERROR] The provided SSH public and private keys for Pattern Manager do not match, please provide a
matching pair of keys.
```

- This error message indicates that the provided pair of private and public keys provided for the Pattern Manager installation do not match. In order to fix it, create a new Content Runtime deployment while making sure that they were generate generated together or, if unsure, generate a new pair of keys.

```
[ERROR] The server's hostname can not contain uppercase letters
```

- This error occurs when the provided host name for the virtual machine contains an uppercase letter and the setup prerequisite checker was set to **strict** mode. When set to **lenient**, the script automatically converts the current host name to lower case. This issue originates from limitations in **Chef**.

```
[ERROR] Docker CE for Red Hat Enterprise is not supported, please provide a valid Docker EE repository
URL
```

- This error is displayed when a RHEL template image is provided without a valid Docker EE repository URL. Docker Community Edition is installed by default on all the other distributions but it does not support RHEL officially.

```
[ERROR] There was an error validating the certificate validation when obtaining Docker Compose from the
provided URL.
```

- When downloading the Docker compose installer, **curl** command is used for obtaining it. This issue is shown when the **curl** command wasn't able to validate the certificate for downloading the file, usually due to a required update in the system. To fix this issue, perform an update on the system and reboot, or skip the certificate verification in **curl** by adding the word **insecure** to **~/.curlrc**. For security reasons, updating the system is the recommended solution path.

```
[ERROR] There is an error with the default permissions (umask) for new users and folders. The
recommended value is 022, found <mask>
```

- The OS image being used contains a UMASK that will limit the required images' permissions when reading or writing files. The default UMASK value can usually be found and changed in the **/etc/login.def** file.

```
[ERROR] Failed while installing <package>
```

- During the prerequisite checking phase of the process, a failure when installing a software package displays this error. If encountered, reviewing the logs can provide more insight on the reason.
- This issue can sometimes be caused by internet connectivity problems or custom setup of repositories which does not include the packages being installed.

VMware vSphere

- The following error is displayed when a virtual machine with the same **name** already exists in the provided VMware vSphere cloud. By default, this **name** is pre-populated to **ibm-content-runtime**. To solve this issue, provide a new unique name for the Content Runtime instance.

```
Error applying plan:
1 error(s) occurred:
* vsphere_virtual_machine.singlenode: 1 error(s) occurred:
* vsphere_virtual_machine.singlenode: The attempted operation cannot be performed in the current state
(Powered on).
```

- The followings error varies according to the setup of the vSphere server in use. Some of the optional fields in the **Optional Cloud Provider Settings** might be required in order to achieve a successful deployment.

```
Error applying plan:
1 error(s) occurred:
* vsphere_virtual_machine.singlenode: 1 error(s) occurred:
* vsphere_virtual_machine.singlenode: default resource pool resolves to multiple instances, please
specify
```

Troubleshooting Docker EE installation

RHEL

The installation of Docker EE requires access to the **container-selinux** package, located by default in the **rhel-7-server-extras-rpms** repository in **/etc/yum.repos.d/redhat.repo**. If the provided Virtual Machine image contains customized RHEL repositories and there is no access to the mentioned **extras-rpms**, the following error is displayed when attempting to install Docker EE.

```
vsphere_virtual_machine.singlenode (remote-exec): Loaded plugins: product-id, search-
vsphere_virtual_machine.singlenode (remote-exec):               : disabled-repos,
vsphere_virtual_machine.singlenode (remote-exec):               : subscription-manager
vphere_virtual_machine.singlenode (remote-exec): Error loading certificate

vsphere_virutal_machine.singlenode (remote-exec): Error getting repository data for rhel-7-server-extras-
rpms, repository not found
vsphere_virtual_machine.singlenode (remote-exec): [ERROR] There was an error installing Docker EE from the
provided repository
vsphere_virtual_machine.singlenode (remote-exec): [ERROR] Repo: https://storebits.docker.com/ee/rhel/sub-b5

Error applying plan:

1 error(s) occurred:

* vsphere_virtual_machine.singlenode: 1 error(s) occurred:

*Script exited with non-zero exit status: 1

Terraform does not automatically rollback in the face of errors. Instead, your Terraform state file has been
partially updated with any resources that successfully completed. Please address the error above and apply
again to incrementally change your infrastructure.
```

Remote installation verification

A verification of the installation can be done by executing the following command:

```
ssh <user>@<content-runtime> advanced-content-runtime/verify-installation.sh
```

The information returned to the screen should help to diagnose the issue.

Known issues

- **CentOS and RHEL**

  The default firewall on the CentOS must be modified to allow for the **docker0** network connection to communicate back to the docker host. The connection **docker0** must be added to the firewall configuration, and port 443 must be enabled for communication.

  ```
  firewall-cmd --permanent --zone=public --change-interface=docker0
  firewall-cmd --permanent --zone=public --add-port=443/tcp
  firewall-cmd --zone=public --list-all
  firewall-cmd --reload
  systemctl status docker
  systemctl restart docker
  docker exec -it camc-pattern-manager /bin/bash -c "nc -v -z <hostip> 443"
  docker exec -it camc-pattern-manager /bin/bash -c "ping google.com"
  ```

  Disabling the firewall causes an issue with the default docker DNS failing to resolve any hosts.

- **localhost.localdomain host name**

  localhost is not supported as a host name for Content Runtime virtual machine. **nmtui** can be used to set the host name on RHEL and CentOS.

- **Mixed case host name**

  The host name of the Content Runtime virtual machine is restricted to lower case. In the case of **other** template, the name is lower cased as part of the Content Runtime configuration process.

- **curl command fails on download of chef-server with SSL certificate error**

```
curl: (60) SSL certificate problem: certificate is not yet valid
More details here: http://curl.haxx.se/docs/sslcerts.html
```

The Content Runtime virtual machine time is too far out of synch with the current time. Set the time of the virtual machine to the current time.

- **Failed connection to external sites**

  The Content Runtime configuration process relies on external sites to download install content for the virtual machine. If a connection to the internet is not available or if one of the sites is down, the Content Runtime installation and configuration process fails. These sites include:

  - Chef
  - Docker Hub
  - GitHub
  - OS update service for the base operating `systemctl`

Health check

If the Content Runtime experiences an unknown issue, read the following sections to know some commands which may help to isolate the problem.

A script on the Content Runtime virtual machine can be executed to display current status for running services.

The script can be executed by running the command:

```
ssh <user>@<content-runtime> advanced-content-runtime/verify-installation.sh
```

where `user` is the user used to create the Content Runtime. The path to the script can be adjusted based on the user to find the correct location of the script.

`content-runtime` is the IP address or host name of the Content Runtime virtual machine.

```
[SUCCESS] Chef server nodes are running correctly
[SUCCESS] Docker is installed
[SUCCESS] Docker is currently running
[SUCCESS] The Pattern Manager image is running correctly
[SUCCESS] The Software Repository image is running correctly
[SUCCESS] Connection to the Software Repo image was established correctly
[SUCCESS] Connection to the Pattern Manager image was established correctly
Connection to 192.168.122.89 443 port [tcp/https] succeeded!
[SUCCESS] Connection from Pattern Manager to host has been established successfully
[SUCCESS] /opt/ibm/docker/software-repo/var/swRepo/private is setup as expected
```

Connection to the Content Runtime virtual machine

The virtual machine running the Content Runtime can be connected using `ssh` and the private key associated with the `User's Public Key` parameter.

Docker

Ensure that the Docker service is running:

```
sudo systemctl status docker
```

or

```
sudo service docker status
```

The service can be started by using the `start` option with the previous command.

If you have installed Docker, you must enable the service to start on reboot. You can check the enablement of the service by running:

```
sudo systemctl is-enabled docker
```

Platforms running `service` should have the service to start by default.

Docker containers

Ensure that the Docker containers are running:

```
sudo docker ps
```

The command output should display two active containers as, for example:

```
CONTAINER ID    IMAGE                                    COMMAND                  CREATED         STATUS
PORTS                                          NAMES
```

```
93e4607e889a    ibmcom/camc-sw-repo:latest            "/bin/bash /tmp/in..."   3 days ago     Up 3 days
0.0.0.0:8888->8888/tcp, 0.0.0.0:9999->9999/tcp    camc-sw-repo
24866ee2426d    ibmcom/camc-pattern-manager:latest    "bin/bash /opt/ibm..."   3 days ago     Up 3 days
0.0.0.0:5443->443/tcp                             camc-pattern-manager
```

If the containers are not running, run the **advanced-content-runtime/launch-docker-compose.sh** command from the ~ directory to restart the containers.

Note that the **docker-compose** command controls stopping and starting the container.

Chef server status

Check the status of the Chef server by using the **/usr/bin/chef-server-ctl status** command.

If there are failed processes, take the appropriate action to recover.

Ubuntu apt-get fails

The configuration of the Content Runtime virtual machine can fail with an **apt-get** lock error. The issue is that a background process holds the lock on the **/var/lib/dpkg/lock** file. The timed upgrade does not complete within a limited amount of time.

```
null_resource.singlenode (remote-exec): Checing lock file /var/lib/dpkg/lock
null_resource.singlenode: Still creating... (18m30s elapsed)
null_resource.singlenode (remote-exec): E: Could not get lock /var/lib/dpkg/lock - open (11: Resource
temporarily unavailable)
null_resource.singlenode (remote-exec): E: Unable to lock the administration directory (/var/lib/dpkg/), is
another process using it?
null_resource.singlenode (remote-exec): [ERROR] This script requires apt-get permissions for executing
```

Ubuntu 16.04 may be configured to run unattended updates. The lock which is held by that process might cause delays to occur with the Content Runtime configuration process.

Disk allocation

The logic within the configuration code takes the largest available non-formatted disk for the Software Repository. If no disk is available, the primary partition is used, with the assumption that it has enough space to hold the Software Repository.

If there is a second disk available, the disk is used as the Docker disk. Usage of the second disk is recommended for production environments.

The configuration code does not support using a separate disk for Docker, but not for the Software Repository.

Disk space

If there is not enough space for the Software Repository, you can allocate new space, and move the content of the Software Repository to that space. Complete this action according to the operating system you selected. When completed, you can run the **advanced-content-runtime/verify-installation.sh** script to determine if the base file structure is correct.

File system

A file directory structure is created for the Software Repository on the **/opt/ibm/docker/software-repo/var/swRepo/** file system.

If you delete or rename any of the file directories, restart the docker container to pick up the changes by running the following command:

```
advanced-content-runtime/launch-docker-compose.sh
```

Managing a software repository

Pattern Management at its core involves the installation and configuration of software packages. In Managed services, the software repository is the node which serves the installation files to the Middleware Template Library installation. The software repository is used to store product binaries that are in compressed format or Installation Manager based repositories.

The software repository currently can hold two different types of product installation files. Product files that are compressed format (zip, tgz) and product binaries that require IBM Installation Manager to perform the installation.

Before you begin

You must have a software repository that is provisioned in the target cloud environment. By default, when the advanced content runtime is provisioned, a software repository is configured.

You must know the following values for the virtual system that is hosting the software repository.

- IP address

    - In the navigation bar, click **Manage > Content Runtimes**.
    - Select the content runtime instance and from the **Actions** overflow menu, click **View Details**
    - In the Overview section, scroll down and click on **Output Variables**.
    - Search for *ibm_sw_repo* and this will contain the IP address of the software repository.
- SSH Key or password to access the virtual system.
- The root directory of the software repository on the virtual system. Default value is **`/opt/ibm/docker/software-repo/var/swRepo/private`**.
- URL for the software repository. For example, **`https://<IP_address>:<port>/`**

Access to the necessary source software repositories from which installation and fix pack files are downloaded:

- IBMid
- Passport Advantage
- Fix Central
- Installation Manager repository

You must have a method to copy files externally to the software repository, either through a direct Internet facing connection or through an appropriate proxy server.

Software Repository Directory Structure

Product binary files must be placed in the correct location within the software repository for the Middleware Template Library content to locate the product binaries. The directory structure is determined by the Enterprise middleware Chef based cookbooks.

The root directory is **`/opt/ibm/docker/software-repo/var/swRepo/private`** and all software binaries reside within this directory structure. The correct directories may be found in the README.md for each of the cookbooks and must be manually created within the software repository. If you do not have access to the middleware Chef cookbooks, see Populating the software repository article for details.

The structure of the repository is as follows:

```
── <product_a>            Name of the software product
    ── <version 1>          Version of the software product
         ── base             Directory that contains the binaries for the full installation of the product.
         ── maint            Directory that contains the fixpack or product updates for the product.
    ── <version 2>
         ── base
         ── maint
── <product b>
    ── <version>
         ── base
         ── maint
── IMRepo                    Directory for Installation Manager binaries
    ── repository.config
```

What to do next

See Populating the software repository for uploading product binaries.

Populating the software repository

Population of the software repository is a manual process which involves creating the directories for each software product and copying the files to these locations.

About this task

This task explains the overall software repository directory structure for products that used compressed format for their installation binaries. For Installation Manager based repositories, see Populating the Installation Manager repository.

Before you begin

1. Understand how to access the advanced content runtime repository so that product binaries can be transferred to the repository.
2. Identify which product binaries need to be transferred to the repository. The product binaries that are needed are based on which Middleware Template are used.
3. Download the product binaries from the appropriate location.

4. Transfer the product binaries to the software repository. See the following diagram for the proper layout.

Software repository directory structure that is properly populated

Product binary files must be placed in the correct location within the software repository for the Middleware Template content to locate the product binaries. The directory structure is determined by the Middleware Template Chef based cookbooks.

The root directory is **`/opt/ibm/docker/software-repo/var/swRepo/private`** and all software binaries reside within this directory structure. For the current supported software products, the software repository directory layout looks like:

- apache > tomcat > v80 > base > apache-tomcat-8.0.15.tar.gz
- db2 > v105 > base > DB2_Svr_10.5.0.8_Linux_x86-64.tar.gz
- db2 > v105 > maint > v10.5fp9_linuxx64_server_t.tar.gz
- db2 > v111 > base > DB2_Svr_11.1_Linux_x86-64.tar.gz
- db2 > v111 > license > DB2_ESE_AUSI_Activation_11.1.zip
- db2 > v111 > maint > v11.1.2fp2_linuxx64_server_t.tar.gz
- im > v1x > base > agent.installer.linux.gtk.x86_64_1.8.6000.20161118_1611.zip
- im > v1x > base > agent.installer.linux.gtk.x86_64_1.8.8000.20171130_1105.zip
- oracle > ee > v12c > base > 12.1.0.2.0 > V46095-01_1of2.zip
- oracle > ee > v12c > base > 12.1.0.2.0 > V46095-01_2of2.zip
- oracle > ee > v12c > base > 12.2.0.1.0 > V839960-01.zip
- oracle > ee > v12c > grid > 12.1.0.1.0 > V38501-01_1of2.zip
- oracle > ee > v12c > grid > 12.1.0.1.0 > V38501-01_2of2.zip
- oracle > ee > v12c > grid > 12.1.0.2.0 > V46096-01_1of2.zip
- oracle > ee > v12c > grid > 12.1.0.2.0 > V46096-01_2of2.zip
- oracle > ee > v12c > grid > 12.2.0.1.0 > V840012-01.zip
- oracle > mysql > v5.7.17 > base > mysql-5.7.17-1.el6.x86_64.rpm-bundle.tar
- oracle > mysql > v5.7.17 > base > mysql-5.7.17-1.el7.x86_64.rpm-bundle.tar
- oracle > mysql > v5.7.17 > base > mysql-server_5.7.17-1debian8_amd64.deb-bundle.tar
- oracle > mysql > v5.7.17 > base > mysql-server_5.7.17-1ubuntu14.04_amd64.deb-bundle.tar
- oracle > mysql > v5.7.17 > base > mysql-server_5.7.17-1ubuntu16.10_amd64.deb-bundle.tar
- wmq > v8.0 > base > WS_MQ_LINUX_ON_X86_64_V8.0_IMG.tar.gz
- wmq > v8.0 > maint > 8.0.0-WS-MQ-LinuxX64-FP0008.tar.gz
- wmq > v9.0 > base > IBM_MQ_9.0.0.0_LINUX_X86-64.tar.gz
- wmq > v9.0 > maint > 9.0.0-IBM-MQ-LinuxX64-FP0002.tar.gz

Downloading IBM software from Passport Advantage and Fix Central

- How to download software from Passport Advantage

IBM WebSphere Application Server

Note: The following links also include details for the IBM HTTP Server and Java SDK.

- Downloading WebSphere Application Server
- Downloading WebSphere Application Server 9.0
- Downloading WebSphere Application Server 8.5.5

IBM DB2 Enterprise Server Edition

- Downloading IBM DB2 11.1
- Downloading IBM DB2 11.5
- How do I download my DB2 license from Passport Advantage
- Recommended Fixes for DB2

IBM MQ

- Downloading IBM MQ 9.0
- Recommended Fixes for IBM MQ

Example: Loading the IBM WebSphere MQ assets into the software repository

After retrieving the product binaries from Passport Advantage (for example):

- IBM MQ V9.0 Long Term Support Release for Linux on x86 64bit Multilingual: **CN9GDML**
- MQ V8.0.0.5 for Multiplatform Multilingual eAssembly: **CJ0CYML**

1. Copy the downloaded files to the software repository:
   - For version 9.0: **`/opt/ibm/docker/software-repo/var/swRepo/private/wmq/v9.0/base`**

- For version 8.0: `/opt/ibm/docker/software-repo/var/swRepo/private/wmq/v8.0/base`

2. Download the available fix packs:
   - `9.0.0-IBM-MQ-LinuxX64-FP0002.tar.gz`
   - `8.0.0-WS-MQ-LinuxX64-FP0008.tar.gz`
3. After you download the latest fixes, copy the downloaded files to the software repository:
   - For version 9.0: `/opt/ibm/docker/software-repo/var/swRepo/private/wmq/v9.0/maint`
   - For version 8.0: `/opt/ibm/docker/software-repo/var/swRepo/private/wmq/v8.0/maint`

Example: Loading the IBM DB2 Enterprise Server Edition into the software repository

After retrieving the product binaries from Passport Advantage:

- IBM® DB2® Server 10.5 for Linux® on AMD64 and Intel® EM64T systems (x64): **CNEB9ML**
- IBM® DB2® Server 11.1 for Linux® on AMD64 and Intel® EM64T systems (x64): **CNB0AML**
- IBM® DB2® Enterprise Server Edition Authorized User Single Install Option Quick Start and Activation V11.1 for Linux, UNIX and Windows Multilingual **CNB23ML**

1. Copy the downloaded files to the software repository:
   - For version 11.0: `/opt/ibm/docker/software-repo/var/swRepo/private/db2/v111/base`
   - For version 10.5: `/opt/ibm/docker/software-repo/var/swRepo/private/db2/v105/base`
2. Download the available fix packs:
   - `v10.5fp9_linuxx64_server_t.tar.gz`
   - `v11.1.2fp2_linuxx64_server_t.tar.gz`
3. After you download the latest fixes, copy the downloaded files to the software repository:
   - For version 10.5: `/opt/ibm/docker/software-repo/var/swRepo/private/db2/v105/maint`
   - For version 11.1: `/opt/ibm/docker/software-repo/var/swRepo/private/db2/v111/maint`
4. Copy the Activation file to the software repository:
   - For version 11.0: `/opt/ibm/docker/software-repo/var/swRepo/private/db2/v111/license`

What to do next

See Populating the Installation Manager repository for setting up the Installation Manager repository for WebSphere Application Server offerings.

Populating the Installation Manager repository

This article explains how to setup the local software repository for products that are based on the IBM WebSphere Application Server family of products. The Installation Manager (IM) repository is served by the content runtime software repository container.

Overview

There are three ways you can populate the Installation Manager repository with the proper WebSphere Application Server offerings.

IBM Installation Manager live repositories

The Installation Manager live repositories enable you to directly install the product and supplemental software over the internet. All fix packs levels are available. Access to the initial release requires the proper entitlement. For a list of Installation Manager repositories, see Online product repositories for WebSphere Application Server offerings.

This method will require you to use the IBM Package Utility to build a local Installation Manager repository.

IBM Fix Central

IBM Fix Central contains all fix packs and interim fixes for the product parts. For example, you can download the latest fix pack for WebSphere Application Server or an earlier level according to your needs. Applicable parts for each version of WebSphere Application Server are listed in the download documents. For a list of updates, see Recommended updates for WebSphere Application Server.

This method allows you to use download the fix packs from IBM Fix Central and copy them to the Advanced Content Runtime software repository. This is a good choice if you already have the fix packs downloaded.

IBM Passport Advantage Online

Passport Advantage customers can download the initial release of the product parts from Passport Advantage Online. To obtain current code levels, these downloads must be supplemented with fix pack repositories from IBM Fix Central. For parts available from Passport Advantage Online, see Download WebSphere Application Server from Passport Advantage Online.

Which method you choose depends on what you are most familiar with.

When working with IBM Installation Manager, you should be familiar with the following terminology:

- **Installation Manager repository**

  A repository that Installation Manager uses to locate the files for installing, modifying, rolling back, updating, or uninstalling WebSphere Application Server products. You specify these repositories in the Installation Manager preferences. When you download a product from IBM Passport Advantage or IBM Fix Central, the extracted installation files contain the repository for the product.

- **Composite repository**

  A repository that references one or more other Installation Manager repositories. By editing this single file, you can quickly and easily change which products and versions the repository includes.

- **External repository**

  A user-created repository that typically exists behind a firewall and can be accessed only by systems within your enterprise's intranet. The external repository can be a composite repository that consists of multiple WebSphere Application Server versions, fix packs, or interim fixes. An external repository is also known as a enterprise repository.

- **IBM Packaging Utility**

  A no-cost companion tool to Installation Manager that you can use to create new repositories that contain one or more WebSphere Application Server product offerings. The tool can accept web repositories or local repositories as input, and it produces a local repository as output. This option is needed if you are using the IBM Installation Manager live repositories.

Procedure

1. Download the Installation Manager toolkit to a local machine and transfer the file to the proper location in the software repository. For more information, see Installation Manager Toolkit
2. Build the Installation Manager repository for the WebSphere Application Server products. For more information, see one of the following:
   - IBM Installation Manager live repositories
   - IBM Fix Central
3. Setup the composite repository. For more information, see Setting up a composite repository
4. Verify that the Installation Manager is setup correctly. For more information, see Validating the Installation Manager repository

What to do next

Deploying a predefined template

Installation Manager toolkit binaries

This article explains how to setup the Installation Manager toolkit product in binaries within the software repository.

Procedure

1. Download the Installation Manager to a local machine. For more information about downloading these files, see Installation Manager and Packaging Utility download documents. See the following table to determine which version of the Installation Manager should be downloaded based on which middleware template version is being used. If you are not sure which version of the cookbooks being used, download all versions of the toolkit.
2. Transfer the Installation Manager toolkit file(s) to the Advanced Content Runtime software repository. The file(s) should be copied to the **`/opt/ibm/docker/software-repo/var/swRepo/private/im/v1x/base`** directory.

Version of Installation Manager Toolkit to download

Installation Manager toolkit 1.8.6 or 1.8.8 are required depending on the version of the cookbooks you are using. Cookbook versions 1.0.2 or earlier require IBM Installation Manager 1.8.6 and versions 1.0.3 or higher require 1.8.8.

The cookbook version, toolkit version, and the corresponding file names are listed in the following table:

Cookbook version	Installation Manager Toolkit version	File Name
1.0.1 or lower	1.8.6	agent.installer.linux.gtk.x86_64_1.8.6000.20161118_1611.zip
1.0.2 or higher	1.8.8	agent.installer.linux.gtk.x86_64_1.8.8000.20171130_1105.zip

What to do next

Continue with the one of the methods for populating the Installation Manger repositories.

Using the Installation Manager live repositories

One of the simplest ways to create an Installation Manager repository that works with WebSphere Application Server family of offerings is to use the IBM Installation Manager live repositories. You can build a single repository that contains a specific WebSphere fix pack version by using the IBM Package Utility.

Before you begin

The Installation Manager toolkit binaries step have been completed.

About this task

Set up the software repository with the proper Installation Manager toolkit and IBM Package Utility that are used to create the Installation Manager enterprise repositories.

Complete the following steps to install the Installation Manager toolkit and the IBM Package Utility product.

Procedure

1. Log in to the Content Runtime virtual machine.
2. Change to the **/opt/ibm/docker/software-repo/var/swRepo/private** directory.

   ```
   cd /opt/ibm/docker/software-repo/var/swRepo/private
   ```

3. Make a directory named **IMTemp**.

   ```
   mkdir IMTemp
   ```

4. Change to the **IMTemp** directory.

   ```
   cd IMTemp
   ```

5. Copy the installer file to the **IMTemp** directory.

   ```
   cp /opt/ibm/docker/software-
   repo/var/swRepo/private/im/v1x/base/agent.installer.linux.gtk.x86_64_1.8.8000.20171130_1105.zip
   /opt/ibm/docker/software-repo/var/swRepo/private/IMTemp
   ```

6. Extract the files from the **agent.installer.linux.gtk.x86_64_1.8.8000.20171130_1105.zip** file.

   ```
   unzip agent.installer.linux.gtk.x86_64_1.8.8000.20171130_1105.zip -d imtoolkit
   ```

7. Change to the **/opt/ibm/docker/software-repo/var/swRepo/private/IMTemp/imtoolkit/tools** directory.

   ```
   cd /opt/ibm/docker/software-repo/var/swRepo/private/IMTemp/imtoolkit/tools
   ```

8. Install the package utility from the IBM public installation manager repository.

   ```
   ./imcl install com.ibm.cic.packagingUtility_1.8.8000.20171130_1136 -repositories
   http://www.ibm.com/software/repositorymanager/com.ibm.cic.packagingUtility  -installationDirectory
   /opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL -dataLocation /opt/ibm/docker/software-
   repo/var/swRepo/private/IMProducts/datalocation -sharedResourcesDirectory /opt/ibm/docker/software-
   repo/var/swRepo/private/IMProducts/IMshared -log /opt/ibm/docker/software-
   repo/var/swRepo/private/IMProducts/IMlogs -prompt -showProgress  -acceptLicense
   ```

9. Change to the IBM Package Utility directory.

   ```
   cd /opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL
   ```

10. Using the IBM Package Utility, build the required Installation Manager repositories based on the product offerings you require. Follow the instructions that follow.

Build the Installation Manager 1.8.6.1 fix pack repository

It is recommended that the Installation Manager 1.8.6.1 be used as it has additional security fixes. This fix pack level is retrieved from the IBM hosted repositories. This is only needed if you are using the Middleware cookbooks version 1.0.2.

1. Copy the Installation Manager 1.8.6.1 fix pack into the repository using the IBM hosted repositories

```
./PUCL copy com.ibm.cic.agent_1.8.6001.20170428_1156 -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.cic.agent  -target /opt/ibm/docker/software-
repo/var/swRepo/private/IMRepo/im186fp1 -prompt  -showProgress -acceptLicense
```

Build the WebSphere v9.0.0.x repository

The commands in the following procedure create a set of installation binaries that supports direct installation to the latest WebSphere Application Server Network Deployment v9.0.0.x fix pack level. If a particular fix pack level is required, the offering name and version number must be specified in the copy command.

The middleware templates are defaulted to install version 9.0.0.6 but they can be changed to use a different fix pack during deployment of the template.

The following table lists the offering IDs that should be used in the Package Utility copy command for various fix pack levels.

Table. WebSphere Application Server

WebSphere Application Server Network Deployment Release	Offering ID
9.0.0.6	com.ibm.websphere.ND.v90_9.0.6.20171205_1311
9.0.0.7	com.ibm.websphere.ND.v90_9.0.7.20180302_0440
latest	com.ibm.websphere.ND.v90

Table. IBM HTTP Server

IBM HTTP Server Release	Offering ID
9.0.0.6	com.ibm.websphere.IHS.v90_9.0.6.20171205_1311
9.0.0.7	com.ibm.websphere.IHS.v90_9.0.7.20180302_0440
latest	com.ibm.websphere.ND.v90

Table. Web server plug-ins

Web server plug-ins Release	Offering ID
9.0.0.6	com.ibm.websphere.PLG.v90_9.0.6.20171205_1311
9.0.0.7	com.ibm.websphere.PLG.v90_9.0.7.20180302_0440
latest	com.ibm.websphere.PLG.v90

Table. Java 8 SDK

Java 8 SDK Release	Offering ID
8.0.5.7	com.ibm.java.jdk.v8_8.0.5007.20171218_1035
8.0.5.10	com.ibm.java.jdk.v8_8.0.5010.20180320_0647
latest	com.ibm.java.jdk.v8

If you do not see the fix pack offering ID, you can use the following command to view the live repositories. You will be prompted for your IBMid.

```
/opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL/PUCL listAvailablePackages -repositories
http://www.ibm.com/software/repositorymanager/V9WASND -prompt
```

The commands in the following procedure build the latest available WebSphere Application Server Network Deployment V9 fix pack level that is available using the IBM hosted repositories. Repeat these commands for each of the fix pack releases that you want to use by replacing the latest offering ID with the appropriate product offering ID.

1. Change your directory to the IBM Package Utility:

```
cd /opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL
```

2. Copy the latest available WebSphere Application Server Network Deployment binaries:

```
./PUCL copy com.ibm.websphere.ND.v90  -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.ND.v90 -target /opt/ibm/docker/software-
repo/var/swRepo/private/IMRepo/WAS9   -prompt -showProgress -acceptLicense
```

3. Copy the latest available IBM HTTP Server binaries:

```
./PUCL copy com.ibm.websphere.IHS.v90 -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.IHS.v90  -target
/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/WAS9   -prompt -showProgress -acceptLicense
```

4. Copy the WebSphere Web Server Plug-ins binaries:

```
./PUCL copy com.ibm.websphere.PLG.v90 -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.PLG.v90/WAS9 -target
```

```
/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/WAS9  -prompt -showProgress -acceptLicense
```

5. Copy the Java 8 SDK binaries:

```
./PUCL copy com.ibm.java.jdk.v8 -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.java.jdk.v8  -target /opt/ibm/docker/software-
repo/var/swRepo/private/IMRepo/jdk8   -prompt  -showProgress -acceptLicense
```

Build the Liberty repository

The commands in the following procedure create a set of installation binaries that supports direct installation to the latest WebSphere Application Server Liberty fix pack level. If a particular fix pack level is required, the offering ID must be used in place of the latest offering ID on the copy command.

The following table lists the WebSphere Liberty core releases and their offering IDs:

Table. WebSphere Liberty Core

WebSphere Liberty Core Release	Offering ID
17.0.0.4	com.ibm.websphere.liberty.CORE_17.0.4.20171206_1447
18.0.0.1	com.ibm.websphere.liberty.CORE_18.0.1.20180309_2209
latest	com.ibm.websphere.liberty.CORE

The following table lists the WebSphere Liberty releases and their offering IDs:

Table. WebSphere Liberty

WebSphere Liberty Release	Offering ID
17.0.0.4	com.ibm.websphere.liberty.BASE_17.0.4.20171206_1447
18.0.0.1	com.ibm.websphere.liberty.BASE_18.0.1.20180309_2209
latest	com.ibm.websphere.liberty.BASE

Table. WebSphere Liberty Network

WebSphere Liberty Network Deployment	Offering ID
17.0.0.4	com.ibm.websphere.liberty.ND_17.0.4.20171206_1447
18.0.0.1	com.ibm.websphere.liberty.ND_18.0.1.20180309_2209
latest	com.ibm.websphere.liberty.ND

The following table lists the Java 8 SDK releases and their offering IDs:

Table. Java 8 SDK

Java 8 SDK Release	Offering ID
8.0.5.7	com.ibm.java.jdk.v8_8.0.5007.20171218_1035
8.0.5.10	com.ibm.java.jdk.v8_8.0.5010.20180320_0647
latest	com.ibm.java.jdk.v8

If you do not see the fix pack offering ID, you can use the following command to view the WebSphere Liberty live repositories. The command prompts for your IBMid.

```
/opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL/PUCL listAvailablePackages -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.CORE -prompt
```

The commands in the following procedure build the latest available WebSphere Application Server Liberty fix pack level that is available using the IBM live repositories. Repeat these commands for each of the fix pack releases that you want to use by replacing the latest offering ID with the appropriate product offering ID.

1. Change your directory to the IBM Package Utility:

```
cd /opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL
```

2. Copy the WebSphere Liberty Core binaries:

```
./PUCL copy com.ibm.websphere.liberty.CORE -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.CORE -target
/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/Liberty   -prompt -showProgress -acceptLicense
```

3. Copy the WebSphere Liberty Network Deployment binaries:

```
./PUCL copy com.ibm.websphere.liberty.ND -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.ND -target
/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/Liberty -prompt  -showProgress -acceptLicense
```

4. Copy the WebSphere Liberty binaries:

```
./PUCL copy com.ibm.websphere.liberty.BASE -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.liberty.BASE -target
/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/Liberty   -prompt   -showProgress -acceptLicense
```

5. Copy the Java 8 SDK binaries:

```
./PUCL copy com.ibm.java.jdk.v8 -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.java.jdk.v8  -target /opt/ibm/docker/software-
repo/var/swRepo/private/IMRepo/jdk8   -prompt  -showProgress -acceptLicense
```

Build WebSphere Application Server 8.5.5.x repository

When building this repository, you need the WebSphere Application Sever 8.5.5.0 Installation Manager repositories in order to build a fully functional repository that is capable of installing the product. You have the following two options:

- Using IBM hosted repositories
- Using IBM hosted repositories and IBM Passport Advantage repositories

If your target repository is already built based upon the 8.5.5.0 code base, you can just use the IBM live repositories.

Using IBM hosted repositories

You can also use the Package Utility **-connnectPassportAdvantage** parameter option to connect to the IBM Passport Advantage repository which allows you to not have to download the V8.5.5.0 product binaries. You must have the appropriate IBMid that is entitled to the WebSphere 8.5.5.0 product binaries.

For example, use the following command to download WebSphere Application Server Network Deployment v8.5.5.12 where your IBMid is entitled to use the IBM Passport Advantage repositories that are IBM hosted:

```
./PUCL copy com.ibm.websphere.ND.v85_8.5.5012.20170627_1018 -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.ND.v85 -target /opt/ibm/docker/software-
repo/var/swRepo/private/IMRepo/WAS8 -showProgress -acceptLicense -prompt -connnectPassportAdvantage
```

Using IBM hosted repositories and IBM Passport Advantage repositories

This option uses a combination of using two source repositories to build a single repository that contains the appropriate fix pack level. This method is used when your IBMid is not entitled to access the IBM Passport Advantage repositories that are configured in the IBM hosted repository site.

For more information, see How to download WebSphere Application Server Network Deployment V8.5.5 from Passport Advantage Online. The required part numbers are listed in the following table:

Note: IBM WebSphere SDK Java Technology Edition 7.1 is not available on the product media or on IBM Passport Advantage. IBM hosted repository can be used to retrieve the Java 7.1 SDK binaries.

After downloading these files, unpackage each product offering file into a separate directory structure, one for the supplements and one for the network deployment. Then use the IBM Package Utility to use two source repositories to create a single new repository. For example:

```
./PUCL copy com.ibm.websphere.ND.v85_8.5.5012.20170627_1018 -repositories
<path_to_8.5.5.0_repostories,http://www.ibm.com/software/repositorymanager/com.ibm.websphere.ND.v85 -target
/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/WAS8 -showProgress -acceptLicense -prompt
```

The following table lists the offering IDs that must be used in the Package Utility copy command for a particular fix pack level.

Table. WebSphere Application Server

WebSphere Application Server Release	Offering ID
8.5.5.12	com.ibm.websphere.ND.v85_8.5.5012.20170627_1018
8.5.5.13	com.ibm.websphere.ND.v85_8.5.5013.20180112_1418
latest	com.ibm.websphere.ND.v85

Table. IBM HTTP Server

IBM HTTP Server Release	Offering ID
8.5.5.12	com.ibm.websphere.IHS.v85_8.5.5012.20170627_1018
8.5.5.13	ccom.ibm.websphere.IHS.v85_8.5.5013.20180112_1418
latest	com.ibm.websphere.IHS.v85

Table. Web server Plug-ins

Web server Plug-ins Release	Offering ID
8.5.5.12	com.ibm.websphere.PLG.v85_8.5.5012.20170627_1018
8.5.5.13	com.ibm.websphere.PLG.v85_8.5.5013.20180112_1418
latest	com.ibm.websphere.PLG.v85

Table. WebSphere Java 7.1 SDK

WebSphere Java 7.1 SDK	Offering ID
8.5.5.12	com.ibm.websphere.IBMJAVA.v71_7.1.4005.20170626_0531
8.5.5.13	com.ibm.websphere.IBMJAVA.v71_7.1.4015.20171219_1604
latest	com.ibm.websphere.IBMJAVA.v71

If you do not see the fix pack offering ID, you can use the following commands to view the WebSphere Liberty live repositories

```
/opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL/PUCL listAvailablePackages -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.ND.v85  -prompt
```

```
/opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL/PUCL listAvailablePackages -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.IBMJAVA.v71  -prompt
```

```
/opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL/PUCL listAvailablePackages -repositories
http://www.ibm.com/software/repositorymanager/com.ibm.websphere.IHS.v85  -prompt
```

The following example illustrates the commands that are used to build a functional WebSphere Application Server Network Deployment repository that is based on the latest fix pack available on IBM hosted repositories:

1. Change your directory to the IBM Package Utility:

   ```
   cd /opt/ibm/docker/software-repo/var/swRepo/private/IMProducts/PUCL
   ```

2. Copy the WebSphere Application Server Network Deployment offering:

   ```
   ./PUCL copy com.ibm.websphere.ND.v85 -repositories
   http://www.ibm.com/software/repositorymanager/com.ibm.websphere.ND.v85 -target /opt/ibm/docker/software-
   repo/var/swRepo/private/IMRepo/WAS8 -showProgress -acceptLicense -prompt
   ```

3. Copy the IBM HTTP Server offering:

   ```
   ./PUCL copy com.ibm.websphere.IHS.v85 -repositories
   http://www.ibm.com/software/repositorymanager/com.ibm.websphere.IHS.v85 -target
   /opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/WAS8 -acceptLicense -prompt
   ```

4. Copy the Web Server Plug-ins offering:

   ```
   ./PUCL copy com.ibm.websphere.PLG -repositories
   http://www.ibm.com/software/repositorymanager/com.ibm.websphere.PLG.v85 -target
   /opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/WAS8 -acceptLicense -prompt
   ```

5. Copy the WebSphere Java 7.1 SDK:

   ```
   ./PUCL copy com.ibm.websphere.IBMJAVA.v71 -repositories
   http://www.ibm.com/software/repositorymanager/com.ibm.websphere.IBMJAVA.v71  -target
   /opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/WAS8 -acceptLicense -prompt
   ```

What to do next

Set up the composite repository. For more information, see Setting up a composite repository.

References

- IBM Packaging Utility Information Center and read about managing packages with Packaging Utility
- Creating custom installation repositories with IBM Packaging Utility for WebSphere Application Server traditional 9
- Creating custom installation repositories with IBM Packaging Utility for WebSphere Application Server 8.5.5

Configuring the Installation Manager repository

WebSphere Application Server family of offerings uses an IBM Installation Manager repository to contain WebSphere Application Server offerings, fix packs, and interim fixes. You can download fix packs and interim fixes from IBM Fix Central and add them to the Advanced Content Runtime software repository composite repository.

Before you begin

The Advanced Content Runtime software repository composite repository contains the WebSphere base product offering that will allow you to perform a full installation of the fix pack. See Download WebSphere Application Server from Passport Advantage Online for instructions.

Procedure

1. Download the appropriate fix pack from IBM Fix Central. See Recommended updates for WebSphere Application Server

   Log in to the Content Runtime virtual machine.

2. Transfer the compressed IBM Installation Manager packages to the /opt/ibm/docker/software-repo/var/swRepo/private/IMRepo directory.

3. Extract the compressed repository files to a directory underneath /opt/ibm/docker/software-repo/var/swRepo/private/IMRepo. Each product offering should be extracted into their own directory.

4. Update the composite repository configuration file **`/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/repository.config`** to contain the new directory.

What to do next

For more information, see Setting up a composite repository.

Setting up a composite repository

A composite repository is one that is reference to one or more other repositories. It allows you to provide one repository to your end users, while being able to quickly and easily change what products and versions that repository includes. The Managed services Installation Manager Repository is set up to be a composite repository.

Product binary files reside in their own directory and have a master **`repository.config`** file that aggregates all of the product binary files. The advantage that this configuration has over a single repository is that you can more easily add or remove product binary files from the overall Installation Manager repository.

The composite repository has the following directory structure:

- **`/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/`**
- **`/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/repository.config`**
- **`/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/<product A>`**
- **`/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/<product B>`**
- **`/opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/<product C>`**

The **`repository.config`** file resembles the following example:

```
#Thu Mar 09 21:59:57 UTC 2017
LayoutPolicyVersion=0.0.0.1
LayoutPolicy=Composite
repository.url.was9=./WAS9
repository.url.was855=./WAS8
repository.url.liberty=./Liberty
repository.url.jdk8=./jdk8
repository.url.imcl186fp1=./im186fp1
```

What to do next

Validate the Installation Manager repository. For more information, see Validating the Installation Manager repository.

Validating the Installation Manager repository

Use the Installation Manager command line and list the contents of the Managed services Installation Manager repository to validate the Installation Manager repository.

1. Change directory to Installation Manager toolkit directory:

   ```
   cd /opt/ibm/docker/software-repo/var/swRepo/private/IMTemp/imtoolkit/tools
   ```

2. List the available packages that are available:

   ```
   ./imcl listAvailablePackages -repositories /opt/ibm/docker/software-repo/var/swRepo/private/IMRepo/
   ```

If the repository is setup properly, you will see output that should contain the WebSphere Application Server offerings that were built using the IBM Package Utility. Here is an example output of a Installation Manager repository that contains the Installation Manager 1.8.6.1, JDK 8 and WebSphere fix pack 9.0.0.4.

com.ibm.cic.agent_1.8.6001.20170428_1156
com.ibm.java.jdk.v8_8.0.5000.20170906_1259
com.ibm.websphere.BASE.v90_9.0.4.20170523_1327
com.ibm.websphere.ND.v90_9.0.4.20170523_1327
com.ibm.websphere.IHS.v90_9.0.4.20170523_1327
com.ibm.websphere.PLG.v90_9.0.4.20170523_1327

Content Runtime Backup Considerations

After a Content Runtime is provisioned, it is recommended customers implement a backup solution.

The following sections will specify the process or file systems that you are recommended to be backed up.

Chef Server Backup

Follow the recommended Chef Server backup and restore steps

- CHEF Backup and Restore

IBM Software Repository Binaries

All the software binaries needed by the Middleware Content Library are located under the file system '**/opt/ibm/docker/software-repo/var/swRepo/private**'.

Customers should backup the file system once populated or have procedure on how to repopulate.

Also backup the software repository configuration under the file system '**/opt/ibm/docker/software-repo/**'.

IBM Pattern Manager

Backup the IBM Pattern Manager configuration under the file system '**/opt/ibm/docker/pattern-manager/**'.

Updating the Content Runtime SSL Certificates

You can update the Chef server certificate on a content runtime deploy. Communication with components of the content runtime is performed with self-signed SSL certificates that are created during deploy. One of these certificates is used by the Chef server for communication with its clients.

The recommended method to replace this certificate with your own certificate is to follow the instructions available at the Chef Security page. Alternatively, you can create a virtual machine and pre-populate the SSL certificate before deploying the content runtime:

1. Create a virtual machine in the cloud.
2. Copy your `.key` and `.pem` files into `/etc/opscode/ca` on the virtual machine.

   Note: If you have a .crt file, ensure that it is in PEM format and rename the suffix to `.pem`.
3. Deploy the type "other" content runtime to your virtual machine.

Middleware Content Templates

After you configure your runtime environments, you can proceed to the content in Middleware Template Library in the Managed services interface to deploy a middleware content. Middleware Content automation are represented in Managed services by the Terraform Template format. These templates are made available through a set of private GitHub repositories.

Before you begin

Read the Managed services Content topic and complete the steps to configure your content runtime environments.

Middleware Template Library

The Middleware Content automation leverages two types of content from this repository:

`template_<template_description>`:

This library repository contains predefined Terraform templates that are used when provisioning a Middleware Template.

This repository contains separate directories for available middleware. For example:

- This repository might be named **`ibm_wasndv9_standalone`**, which contains the templates for a WebSphere standalone topology.
- This **`ibm_wasndv9_standalone`** repository contains these child directories: **`amazon`**, **`ibmcloud`**, and **`vmware`**.
- These child directories contain a **`terraform`** directory, which contains the template for the WebSphere Application Server deployment.
- This directory also contains the **`camvariables.json`** file, which is the parameters file that is used to customize the deployment experience.

`cookbook_<middleware_product>`:

These repositories contain the Chef based automation that is used for the installation and configuration of the corresponding software.

- The **`cookbook_<middleware_product>`** directory contains the required files: **`README.md`**, **`attributes.json`**, **`components.json`**, **`metadata.rb`**, and **`recipes.json`**.
- It also contains several subdirectories which contain the required files for the cookbook:
  - **`attributes`**: This directory contains the attributes files: **`default.rb`** and **`internal.rb`**.
  - **`libraries`**: This directory contains chef libraries to extend the cookbook functionality.
  - **`providers`**: This directory contains cookbook Lightweight Resource Providers (LWRPs) to extend cookbook functionality
  - **`recipes`**: This directory contains the recipes for the cookbook, such as **`prereq.rb`**, **`install.rb`**, **`gather_evidence.rb`**, **`fixpack.rb`**, **`harden.rb`**, and more.
  - **`resources`**: This directory contains resources to be used with the LWRPs under the provider directory.

Chef cookbooks

IBM-created Chef cookbooks use a common structure to provide a consistent experience. Each cookbook repository holds the automation for a single product. When possible, a cookbook will support multiple versions of a product offering, as well as support for multiple operating systems.

Cookbook and Template Versions

Currently templates supported and versions, see IBM-CAMHub-Open/IBM-CAMHub-Open repository.

What's Next

Proceed to Deploying a Middleware template to know more about deploying a middleware template.

Proceed to these section if you want to know more about middleware templates in GitHub repositories and and how it relates to relates to Terraform.

- Terraform Template Overview
- Template Cloud Parameters
- Template Middleware Parameters

Deploying a middleware templates

You can deploy the predefined middleware templates in GitHub repository by first importing them using the steps in Import page. Once you import, you can then deploy them to your cloud environment including VMware, IBM Cloud, and AWS.

Starting from IBM Cloud Pak® for AIOps 4.2.1 release, terraform engine 0.11.x is not shipped with the product. The middleware content requires terraform engine 0.11.x. You must use the Create a version steps to first import the 0.11.x terraform engine before you deploy middleware templates.

General deployment guidelines

- Each predefined template was tested with the default values for its variables. See the README file for the template for details on possible customizations to the template.

Deployment guidelines for VMware

- Consult the README file for the template to learn what additional configuration steps might be required to deploy that template.
- No additional general configuration steps are required for VMware.

Deploy the template

For steps to deploy the template, see Deploying a WebSphere Application Server V9 Standalone template.

Validate the deployment

After you deploy a predefined template, you can validate the deployment by checking the log files for Terraform, Chef, and the middleware product. You can also check the Output variables tab of the deployed template and confirm you see the desired output.

Troubleshooting

For details about troubleshooting issues when deploying templates, see Troubleshooting content related issues and useful links.

Deploying a WebSphere Application Server V9 Standalone template

Deploy the WebSphere Application Server V9 Standalone template in your environment.

Starting from IBM Cloud Pak® for AIOps 4.2.1 release, terraform engine 0.11.x is not shipped with the product. This template requires terraform engine 0.11.x. You must use the Create a version steps to first import the 0.11.x terraform engine before you deploy this template.

Procedure

To deploy the WebSphere Application Server V9 Standalone template, follow these steps:

1. In the navigation bar, click **Library** > **Terraform templates**. The template library is displayed.
2. Import the template from GitHub repository
3. Click **Middleware**.
4. Click **IBM WebSphere Network Deployment V9 on a single virtual machine - VMware vSphere**.
5. Read the information about the template and click **Deploy template**.
6. Enter the required parameters:
   - Specify the **Instance name**.
   - Select the **VMware** cloud connection.
   - Select the **Advanced content runtime** Namespace.
7. Specify the instance name, a connection to the cloud provider where you want to deploy the template, and enter the required deployment parameters.

   The cloud connection list only contains the connections defined for the type of cloud provider where you can deploy the template. For information about cloud connections, see Managing connections.
8. Click **Deploy**. The template instance window is displayed.

   In the template instance window, you can see the instance status, the instance information, the related resource details, and the related input and output variables. You can view or copy the instance logs and you can also access the template from which the instance was deployed.

Cloud Input Variables Group

Table. Cloud Input Variables Group

Parameter	Default Value	Description
vSphere Cluster		The name of the target vSphere cluster to host virtual machine.
vSphere Datacenter		The name of the target vSphere Datacenter to host virtual machine.
vSphere Folder		The name of the target vSphere folder to host virtual machine.

Virtual Machine Input Variables Group

Table. Virtual Machine Input Variables Group

Parameter	Default Value	Description
DNS servers		List of DNS servers for the virtual network adapter. For more than one DNS, the values can be entered by adding more fields.
DNS Suffixes		List of DNS Suffixes for the virtual network adapter. For more than one DNS, the values can be entered by adding more fields.
Domain Name		Virtual Machines domain name
Operating System ID/Template		The vSphere Template Name to be cloned
Operating System Username	root	The username to log into the Template System ID / Template
Operating System Password		The password for the Operating System Username to log into the Template System ID/Template
Short Hostname		The short hostname of the virtual machine
Template Disk Controller	scsi	A drop-down of the vSphere template disk controller types.
Template Disk Datastore		Data store or storage cluster name for target virtual machine's disks
Template Disk Type	Thick Provision Eager-zeroeed	A drop-down list of vSpghere Disk types
Virtual Machine Gateway Address		IPv4 gateway for vNIC configuration
Virtual Machine IP Address		IPv4 address for vNIC configuration
Virtual Machine Memory		Memory assigned to the virtual machine in megabytes. This value is required to be an increment of 1024
Virtual Machine Netmask Prefix		IPv4 prefix length for vNIC configuration. The value must be a number between 8 and 32
Virtual Machine vCPU's		Number of virtual CPU for the virtual machine, which is required to be a positive Integer
Virtual Machine vSphere Port Group		vSphere port group or network label for virtual machine's vNIC

SSH Keys Group

Table. SSH Keys Group

Parameter	Default Value	Description
User Public SSH Key		User defined public SSH key used to connect to the virtual machine. The format must be in openSSH.

WASNode01 Group

Table. WASNode01 Group

Parameter	Default Value	Description
WebSphere Application Server Version	9.0.0.4	The release and fix pack level of WebSphere Application Server to be installed
WebSphere Application Server maximum JVM Heap Size	512	WebSphere Application Server maximum JVM Heap Size
WebSphere Application Server minimum JVM Heap Size	256	WebSphere Application Server minimum JVM Heap Size
WebSphere Application Server profile name	standalone1	WebSphere Application Server Profile name
WebSphere Java SDK Version	8.0.4.70	The Java SDK version that should be installed with the WebSphere Application Server
WebSphere administrative user name	wasadmin	The username for securing the WebSphere administrative console
WebSphere administrative user password	xxxx	The password for the WebSphere administrative account
WebSphere default keystore password	xxxx	Specifies the password to use on all keystore files created during profile creation
WebSphere product installation directory	/opt/IBM/WebSphere/AppServer	The installation root directory for the WebSphere Application Server product binaries"
WebSphere profile location	/opt/IBM/WebSphere/AppServer/profiles	The directory path that contains WebSphere Application Server profiles

Deploying content and content runtime from Managed services that is behind a firewall

Deployment of content and content runtime in Managed services that is behind a firewall.

About the task

Two scenarios are supported for this deployment:

- Deploy the content and content runtime on to a cloud that is outside the firewall. For example, IBM Cloud, AWS and vSphere from the Managed services that is inside the firewall. For this deployment, you need a Bastion Server that is capable of SSH calls. The Terraform template uses this server to make any remote execution calls.
- Deploy the content and content runtime to a vSphere cloud that is inside a firewall from the Managed services that is also inside a firewall. For this deployment, you need a HTTP proxy server that can forward any HTTP or HTTPS calls. The limitations for this scenario are as follows:
  - For middleware content, support is available only for Ubuntu.
  - For Content Runtime, Bring Your Own Chef is not supported.
  - Starter library MEAN stack template is not supported.

Procedure

1. In Managed services, go to **Manage** > **Shared parameters**.
2. You can view the shared parameters that are included to support the scenarios mentioned in the "About the task" section.
   - The following Data Types are created to capture Bastion host details and HTTP configuration details:
     - `Bastion Host Configuration` - Configuration that contains information to connect outbound through a Bastion host.
     - `HTTP Proxy Configuration`- Proxy configuration that is to be used for both HTTP and HTTPS outbound communication by the virtual machine that is deployed inside a proxy.
   - The following Data Objects that are provided can be used whenever you do not need a Bastion host or HTTP proxy host:
     - `DefaultNoBastionHostRequired`
     - `DefaultNoProxyRequired`
3. If you need a Bastion host or HTTP proxy, then you must create your own Data Objects that capture the details of the Bastion host or HTTP proxy server. For steps to create Data Types and Data Objects, see Managing shared parameters from IBM Managed services console.
4. Open a Content Runtime, Terraform template, or middleware template to deploy:
   - To deploy from the Managed services that is not behind a firewall and also wherein both Bastion host and HTTP proxy are not required:
     - Select the cloud connection.
     - In the **Select data object** section, select `DefaultNoBastionHostRequired` for Bastion Host Configuration.
     - Select `DefaultNoProxyRequired` for **HTTP Proxy Configuration**.
   - To deploy to a public cloud from Managed services that is behind a firewall:
     - Select the cloud connection.
     - In the **Select data object** section, for **Bastion Host Configuration**, select the Bastion host data object that encapsulates your bastion host data.
     - Select `DefaultNoProxyRequired` for **HTTP Proxy Configuration**.
   - Deploy Content Runtime, middleware, and other templates on to a private VMware cloud that is behind a firewall from Managed services that is also behind a firewall: To deploy to a private VMware, additional HTTP Proxy Configuration details are required.
     - Select the cloud connection.
     - In the **Select data object** section, select `DefaultNoBastionHostRequired` for **Bastion Host Configuration**.
     - For **HTTP Proxy Configuration**, select the **HTTP Proxy Configuration** data object that encapsulates your HTTP proxy server.
5. Click **Deploy**.

Terraform Template Overview

The following is a description of a typical Terraform template focusing on the elements which may be modified.

The following elements are considered:

- Variables - SSH keys
- Variables - Cloud specific

- Variables - Environment variables
- Variables - Pattern variables
- Variables - Image variables
- IBM-specific Terraform resources

Variables - SSH keys

SSH keys for injection and Pattern Manager remote control.

```
variable "user_public_ssh_key" {
  type = "string"
  description = "User defined public SSH key used to connect to the virtual machine, whose format is required
to be the openSSH key format with optional comment"
  default = "None"
}
```

Variables - Cloud specific

Cloud specific variables that may be different for each cloud type. Cloud specific variables are likely injected by Managed services and not exposed to the deployer. The following examples are for VMware-based cloud.

```
variable "vsphere_user" {
  description = "vSphere user"
}

variable "vsphere_password" {
  description = "vSphere password"
}

variable "vsphere_server" {
  description = "vSphere server"
}

variable "ibm_pm_public_ssh_key" {
  description = "Public CAMC SSH key value which is used to connect to a guest, used on VMware only"
}

variable "ibm_pm_private_ssh_key" {
  description = "Private CAMC SSH key (base64 encoded) used to connect to the virtual guest"
}

variable "allow_unverified_ssl" {
  description = "Communication with vSphere server with self-signed certificate"
  default = "true"
}
```

Variables - Environment variables

Environment variables relate to the injected variables which relate to the selected Content Runtime, and are not exposed to the deployer. A typical set of variables is:

```
variable "vsphere_user" {
  description = "vSphere user"
}

variable "vsphere_password" {
  description = "vSphere password"
}

variable "vsphere_server" {
  description = "vSphere server"
}

variable "ibm_pm_public_ssh_key" {
  description = "Public CAMC SSH key value which is used to connect to a guest, used on VMware only"
}

variable "ibm_pm_private_ssh_key" {
  description = "Private CAMC SSH key (base64 encoded) used to connect to the virtual guest"
}

variable "allow_unverified_ssl" {
  description = "Communication with vSphere server with self-signed certificate"
  default = "true"
```

```
}
```

Variables - Pattern variables

This set of variables contains the specific variables that relate to the software pattern. Variables are typically defined on a per-node basis and relate to specific software concepts such as ports, users and directories.

This type of variable is the most applicable for editing. You can create custom templates for specific situations and hide variables from the deployer if necessary. It is noteworthy that each of these parameters appears in the CAM Variables section and may be defaulted or hidden by CAM Variable modification.

```
##### centralnode variables #####

#Variable : centralnode_was_liberty_base_version
variable "centralnode_was_liberty_base_version" {
  type = "string"
  description = "The release and fixpack level for WebSphere Liberty to be installed. Example formats are
8.5.5.11 or 17.0.0.2"
  default = "17.0.0.2"
}

#Variable : centralnode_was_liberty_edition
variable "centralnode_was_liberty_edition" {
  type = "string"
  description = "Indicates which Liberty offering should be installed. Valid values are: base, core, nd"
  default = "base"
}
```

Variables - Image variables

A template describes one or more compute instances and other cloud resources. These resources are cloud specific and may include the definition of virtual machines, networks, disk or security groups. The templates have different resources for each cloud defined.

This category of variables may be extremely relevant for an administrator to modify, to ensure that template deployers do not have too many decisions to make at deploy time.

```
#########################################################
##### Resource : centralnode
#########################################################

variable "centralnode-os_password" {
  type = "string"
  description = "Operating System Password for the Operating System User to access virtual machine"
}

variable "centralnode_folder" {
  description = "Target vSphere folder for virtual machine"
}

variable "centralnode_datacenter" {
  description = "Target vSphere Datacenter for virtual machine creation"
}

variable "centralnode_number_of_vcpu" {
  description = "Number of virtual CPU for the virtual machine, which is required to be a positive Integer"
  default = "2"
}
```

It is noteworthy that these resources often use CAM Variables to set values at deploy time. For example, the amount of memory assigned to a virtual machine is defined as follows and may be modified by an administrator using CAM Variables.

IBM-specific Terraform resources

IBM has extended the standard Terraform provider to expose methods provided by the Pattern Manager which participate in software deployment. The following resources are exposed:

camc_bootstrap

`camc_bootstrap` bootstraps the target node with a Chef client and register the client with the relevant Chef server. A Chef vault item is also created at this time to handle Pattern Secrets. Standard IBM Variables such as **`stack_id`** and **`stack_name`** are created as Node Attributes during **`camc_bootstrap`**. An example of **`camc_bootstrap`** is:

```
resource "camc_bootstrap" "centralnode_chef_bootstrap_comp" {
  depends_on = ["camc_vaultitem.VaultItem","vsphere_virtual_machine.centralnode"]
  name = "centralnode_chef_bootstrap_comp"
  camc_endpoint = "${var.ibm_pm_service}/v1/bootstrap/chef"
  access_token = "${var.ibm_pm_access_token}"
  skip_ssl_verify = true
  trace = true
  data = <<EOT
{
  "os_admin_user": "${var.centralnode-os_admin_user}",
  "stack_id": "${random_id.stack_id.hex}",
  "environment_name": "_default",
  "host_ip": "${vsphere_virtual_machine.centralnode.network_interface.0.ipv4_address}",
  "node_name": "${var.centralnode-name}",
  "node_attributes": {
    "ibm_internal": {
      "stack_id": "${random_id.stack_id.hex}",
      "stack_name": "${var.ibm_stack_name}",
      "vault": {
        "item": "secrets",
        "name": "${random_id.stack_id.hex}"
      }
    }
  }
}
EOT
}
```

camc_softwaredeploy

camc_softwaredeploy instructs the Pattern Manager to execute a Chef role on the target node. An important facet of this operation is the creation of Node Attributes on the Chef node which are used when the Chef client is executed.

```
resource "camc_softwaredeploy" "centralnode_liberty_install" {
  depends_on =
["camc_bootstrap.centralnode_chef_bootstrap_comp","camc_bootstrap.libertynode_chef_bootstrap_comp"]
  name = "centralnode_liberty_install"
  camc_endpoint = "${var.ibm_pm_service}/v1/software_deployment/chef"
  access_token = "${var.ibm_pm_access_token}"
  skip_ssl_verify = true
  trace = true
  data = <<EOT
{
  "os_admin_user": "${var.centralnode-os_admin_user}",
  "stack_id": "${random_id.stack_id.hex}",
  "environment_name": "_default",
  "host_ip": "${vsphere_virtual_machine.centralnode.network_interface.0.ipv4_address}",
  "node_name": "${var.centralnode-name}",
  "runlist": "role[liberty_install]",
  "node_attributes": {
    "ibm": {
      "im_repo": "${var.ibm_im_repo}",
      "im_repo_user": "${var.ibm_im_repo_user}",
      "sw_repo": "${var.ibm_sw_repo}",
      "sw_repo_user": "${var.ibm_sw_repo_user}"
    },
    "ibm_internal": {
      "roles": "[liberty_install]"
    },
    "was_liberty": {
      "base_version": "${var.centralnode_was_liberty_base_version}",
      "edition": "${var.centralnode_was_liberty_edition}",
      "im_install_dir": "${var.centralnode_was_liberty_im_install_dir}",
      "install_dir": "${var.centralnode_was_liberty_install_dir}",
      "install_grp": "${var.centralnode_was_liberty_install_grp}",
      "install_user": "${var.centralnode_was_liberty_install_user}",
      "java_version": "${var.centralnode_was_liberty_java_version}",
      "wlp_user_dir": "${var.centralnode_was_liberty_wlp_user_dir}"
    }
  },
  "vault_content": {
    "item": "secrets",
    "values": {
      "ibm": {
        "im_repo_password": "${var.ibm_im_repo_password}",
        "sw_repo_password": "${var.ibm_sw_repo_password}"
      }
    },
    "vault": "${random_id.stack_id.hex}"
  }
}
```

```
EOT
}
```

In the previous example, note the following items:

- `runlist`: The Chef Runlist to execute.
- `host_ip`: The IP address of the node to execute the Chef client on.
- `node_attributes`: A Chef structure list of node attributes to be created.
- `vault_content`: Chef vault items to create.

  Note: A single Chef vault is created per template or stack. In this way, all nodes from the same template can share secrets and exclude nodes from other templates.

Template Cloud Parameters

The following sections detail common and standard template parameters and how they are used in Managed services. Note that the predominant classification is by cloud type, which reflects the differences among cloud types when provisioning cloud resources.

Global variables

Global variables appear once on a Terraform template and effect the template as a whole. Templates that deploy to specific clouds may have variables unique to that cloud type.

Table 1. Global Variables

Attribute	Description	Scope
ibm_stack_name	A unique stack name which is appended as a Chef Node Attribute. System generated.	all
allow_unverified_ssl	Boolean. Communication with vSphere server with self-signed certificate.	VMware
client_debug	True or False. If you enable it to true in the provider section, the logs are written to `/home/terraform/.govmomi/debug` directory.	VMware
aws_ami_owner_id	The AMI owner ID, injected by the cloud connection.	AWS
aws_region	The AWS region, injected by the cloud connection. For example, us-east-1.	AWS
aws_sg_camc_name	The name of the AWS security group for automation content, injected by the cloud connection.	AWS
aws_vpc_name	The name of the AWS VPC, injected by the cloud connection.	AWS

Content Runtime injected variables

The Content Runtime predefines a specific set of variables that are injected to the template at deployment time. These variables can not be modified in CAM Variables. They are listed as a reference only.

Table 2. Content Runtime Injected Variables

Attribute	Description	Scope
ibm_sw_repo	IBM Software Repository root, described by a URL.	all
ibm_sw_repo_password	IBM Software Repository password, set by the HTTP Server on the software repository.	all
ibm_sw_repo_user	IBM Software Repository user name, used to secure the HTTP Server.	all
ibm_pm_private_ssh_key	Private CAMC SSH key (base64 encoded) used to connect to the virtual guest.	all
ibm_pm_public_ssh_key	Public CAMC SSH key name used to connect to the virtual guest.	all
ibm_pm_access_token	IBM Pattern Manager access token.	all
ibm_pm_service	IBM Pattern Manager service.	all

Node variables - All clouds

Node variables appear once per node and can therefore appear multiple times on the same template. Node variables also have a name which is a merging of the variable name and the node name. For example, the os_admin_user field may appear in the template as MQV9Node01-os_admin_user, which is a combination of the variable and node name.

Table 3. Node Variables - All Clouds

Attribute	Description	Scope
domain	Domain Name of virtual machine. It is used in DNS searching on the node (resolv.conf on Linux devices).	all
runtime_domain	Domain Name of virtual machine. It is used in DNS searching on the node (resolv.conf on Linux devices).	all
image	Operating system image ID/template that should be used when creating the virtual image.	all
name	Short host name of the virtual machine.	all
user_public_ssh_key	User-defined public SSH key used to connect to the virtual machine. Set to None if not required.	all

Node variables - VMware

The following is a list of node parameters that effect VMware-based templates only.

Table 4. Node Variables - VMware

Attribute	Description	Scope
os_password	Operating system password for the operating system user to access virtual machine, used by Terraform to bootstrap the node.	VMware
root_disk_controller_type	Type of template disk controller. For example, scsi-lsi-parallel.	VMware
root_disk_datastore	Data store or storage cluster name for target virtual machine's disks.	VMware
root_disk_type	Type of template disk volume. For example, eager_zeroed.	VMware
ipv4_gateway	IPv4 gateway for vNIC configuration.	VMware
ipv4_address	IPv4 address for vNIC configuration.	VMware
memory	Memory assigned to the virtual machine in megabytes. This value is required to be an increment of 1024.	VMware
ipv4_prefix_length	IPv4 prefix length for vNIC configuration. The value must be a number between 8 and 32.	VMware
number_of_vcpu	Number of virtual CPU for the virtual machine. The value must be a positive integer.	VMware
folder	Target vSphere folder for virtual machine.	VMware
datacenter	Target vSphere datacenter for virtual machine creation.	VMware
cluster	Data store or storage cluster name for target virtual machine's disks.	VMware
network_interface_label	vSphere port group or network label for virtual machine's vNIC.	VMware

Node variables - AWS

The following table lists node parameters that effect AWS-based templates only.

Table 5. Node Variables - AWS

Attribute	Description	Scope
root_block_device_delete_on_termination	Boolean. AWS root block device delete on termination.	AWS
root_block_device_volume_size	AWS root block device volume size in gigabytes.	AWS
root_block_device_volume_type	AWS root block device volume type. For example: gp2, io1, st1, sc1.	AWS
subnet_name	AWS subnet name.	AWS
associate_public_ip_address	Boolean. Assign a public IP address.	AWS
flavor	AWS flavor of the virtual machine. For example, t2.small.	AWS
mgmt-network-public	Boolean. Expose and use public IP address of virtual machine for internal communication.	AWS

Node variables - IBM Cloud

The following table lists node parameters that effect IBM Cloud-based templates only.

Table 6. Node Variables - IBM Cloud

Attribute	Description	Scope
local_disk	Boolean. User local disk or SAN disk.	ibm
datacenter	IBM Cloud datacenter where infrastructure resources is deployed.	ibm
dedicated_acct_host_only	Shared or dedicated host, where dedicated host usually means higher performance and cost.	ibm
hourly_billing	Billing cycle: hourly billed or monthly billed	ibm
network_speed	Bandwidth of network communication applied to the virtual machine. For example, 10.	ibm

Attribute	Description	Scope
private_network_only	Boolean. Provision the virtual machine with only private IP address.	ibm
memory	Amount of Memory (MBs), which is required to be one or more times of 1024.	ibm
number_of_cores	Number of CPU cores, which is required to be a positive integer.	ibm
root_disk_size	Root disk size in GB. For example, 25.	ibm
mgmt-network-public	Boolean. Expose and use public IP address of virtual machine for internal communication.	ibm

Template Middleware Parameters

Managed services variables are definitions of variables that are presented to the deployer when a Template is deployed. This section will describe the format and uses of these variables.

Managed services variables provide the opportunity for an administrator to hardcode certain values for a parameter and make them relevant for a specific organization. Additionally, the deployment properties of variables may be modified to make it suitable for a specific situation.

Managed services variables may be accessed and edited from the **Templates** tab in the Template Editor.

From GIT, the *camvariables.json* file describes a set of Managed services variables for a specific template.

Managed services Variables example

The following example shows a sample Managed services variable as it is described in the **Parameters** section of the Template Editor. To understand the context of this variable, the Terraform template must be inspected.

```
python
  {
    "name": "referenced-image-image",
    "type": "string",
    "description": "referenced-image OS Image",
    "default": "RHEL-7.3_HVM_GA",
    "hidden": false,
    "label": "referenced-image Image",
    "secured": false,
    "system": false,
    "required": true,
    "immutable": false
  }
```

This example describes an input variable to a **Terraform Template** that gets interpreted by Managed services when the template is deployed.

Managed services variables structure

The following is the meta-structure of a Managed services variables file, this in general represents the different sections contained within the file.

Section	Definition
input_datatypes	input_datatypes are a collection of attributes. This section defines one or more input_datatypes collections to inject to the template at deploy time the most common being advanced_content_runtime_chef. <\br> You can also use input_datatypes to define Data Types.
output_datatypes	After deployment, the this section defines the name of the namespace to be updated post deployment.
input_groups	Variables are divided into logical groups. One such use of variable groups is to more efficiently display variables to a deployer in a logical format. This section defines one or more Input Groups associated with a Template.
output_groups	This section defines one or more Output Groups associated with a Template.
template_input_params	Templates will accept one or more parameters in order to deploy. As a general rule, these parameters will help populate the Terraform Variables (.tfvars) file. The input parameters define a list of parameters to be inputted to the Terraform Template during the deployment process. The field definition will dictate how CAM handles these variables at deploy time. <\br> If you want to get a value from datatype, then you can define the default value, for example, "default": "${datatypedemo1.ipaddress}", where datatypedemo1 is the name of the datatype and ipaddress is the attribute.
template_output_params	A list of variables calculated during the deployment process and stored for future reference.

Input Parameter Definition

The following are the relevant field definitions for defining Input Parameters:

Field	Definition
name	The name of the service parameter.
label	The label to be displayed in the user interface for this parameter
description	A description for this parameter.
type	The type of parameter. Such as "list, string, map.
default	The default value of the parameter. An example default value for shared parameter is `"default": "${datatypedemo1.ipaddress}"`, where `datatypedemo1` is the name of the Data Type and `ipaddress` is an attribute.
hidden	Specify if the parameter is not displayed in the user interface. The value must be true or false.
immutable	Specify if the parameter is read only. The value must be true or false.
immutable_after_create	Specify a value and deploy it for the first time. You cannot edit it in subsequent plan changes. The value must be true or false.
required	Specify if the parameter is required. The value must be true or false.
secured	Specify if the parameter value is shown as ********. The value must be true or false.
regex	Define the regular expression (regex) - A-Za-z0-9-[1]{1,52}$. The value you provide must match this regular expression.
group_name	The group to locate this parameter in when displaying on the UI" Such as "myGroup".
options	in place of "default" for a list to select from

Output Parameter Definition

The following are the relevant field definitions for defining Output Parameters:

Field	Definition
name	The name of the service parameter.
label	The label to be displayed in the user interface for this parameter
description	A description for this parameter.
type	The type of parameter. Such as "list, string, map
hidden	Specify if the parameter is not displayed in the user interface. The value must be true or false.
secured	Specify if the parameter value is shown as ********. The value must be true or false.
return_type	The type of the parameter that is returned.
group_name	The group to locate this parameter in when displaying on the UI" Such as "myGroup".

Relationship to templates

Generally, Managed services variables directly relate to the parameters exposed in a Terraform template. Managed services variables are used by Managed services to inject Terraform parameters at deploy time.

---

1. A-Za-z0-9- ↩

Customizing a predefined template

The Managed services Middleware templates can be customized to fit your organizational requirements. Updating the templates is a multi-step process which involves cloning the private GitHub repositories and modifying the template input parameters.

Follow these steps to customize the Middleware templates:

1. Clone the Middleware templates.
2. Update template parameters file.
3. Modify Template.
4. Modify CAM Variables.

What to do next

Deploy a Middleware template.

Template modification

Template modification is possible by cloning catalog items from within the Managed services user interface and directly modifying the Template inline.

Consider the following use cases:

- Hardcoding template parameters
- Template modification - BYO script
- Template modification - BYO Chef role

Hardcoding template parameters

At a template level, you can hardcode predefined parameters and set them within the template. The following changes must be made:

- Remove Terraform parameter.
- Hardcode value in the **camc_softwaredeploy** resource.

As an example, consider the following **camc_softwaredeploy** resource and associated parameter, **install_dir**:

```
  #Variable : centralnode_was_liberty_install_dir
  variable "centralnode_was_liberty_install_dir" {
    type = "string"
    description = "Liberty install dir"
    default = "/opt/IBM/WebSphere/Liberty"
  }

  resource "camc_softwaredeploy" "centralnode_liberty_install" {
  ...

    "runlist": "role[liberty_install]",
    "node_attributes": {
      "was_liberty": {
        "install_dir": "${var.centralnode_was_liberty_install_dir}",

  ...
```

In the previous example, if you want to hardcode the value of the **install_dir** parameter, you must first remove the variable and secondly modify the resource with the correct value as, for example, **/opt/liberty**.

```
  "node_attributes": {
    "was_liberty": {
      "install_dir": "/opt/liberty",
```

Subsequent deployments of the template ignore parameters injected, and instead always use your custom value.

Template modification - BYO script

The BYO script resource allows for the inclusion of a custom script into an existing template. To perform this modification, the Managed services user interface may be used.

Note that the addition of custom scripts uses the standard Terraform **remote-exec** resource. The following sample shows how this may be used with standard IBM-provided variables.

- **Custom script snippet**

  Use the following Terraform snippet in a VMware based template to add a custom script to the template.

```
##############################################################
##### Resource : Sample Script
##############################################################

resource "null_resource" "NODENAME_Sample Script" {
  depends_on = ["vsphere_virtual_machine.NODENAME"]

  # Specify the ssh connection
  connection {
    user = "${var.NODENAME-os_admin_user == "" ? lookup(var.default_os_admin_user, var.NODENAME-image) :
var.NODENAME-os_admin_user}"
    private_key = "${base64decode(var.ibm_pm_private_ssh_key)}"
    host = "${vsphere_virtual_machine.NODENAME.network_interface.0.ipv4_address}"
  }

  provisioner "file" {
    destination = "NODENAME_Sample Script.properties"
    content     = <<EOF
export custom_property=""
```

```
EOF
  }

  provisioner "file" {
    destination = "NODENAME_Sample Script.sh"
    content     = <<EOF
#!/bin/bash

++SCRIPTLINES++
EOF

  # Execute the script remotely
  provisioner "remote-exec" {
    inline = [
      "sudo bash -c 'chmod +x NODENAME_Sample Script.properties'",
      "sudo bash -c '. ./NODENAME_Sample Script.properties'",
      "sudo bash -c 'chmod +x NODENAME_Sample Script.sh'",
      "sudo bash -c './NODENAME_Sample Script.sh  >> centralnode_Sample Script.log 2>&1'",
      "sudo bash -c 'rm NODENAME_Sample Script.properties'"
    ]
  }

}
```

The following variables must be substituted:

- **NODENAME**: The logical name of the node within the Terraform template.
- **SCRIPTLINES**: The injected script.

Template modification - BYO Chef role

The BYO Chef role provides the capability to insert a customer provided Chef role into the orchestration. The following snippet must be added to a Terraform template:

```
resource "camc_softwaredeploy" "NODENAME_ROLE" {
  depends_on = ["vsphere_virtual_machine.NODENAME"]
  name = "NODENAME_ROLE"
  camc_endpoint = "${var.ibm_pm_service}/v1/software_deployment/chef"
  access_token = "${var.ibm_pm_access_token}"
  skip_ssl_verify = true
  trace = true
  data = <<EOT
{
  "os_admin_user": "${var.NODENAME-os_admin_user}",
  "stack_id": "${random_id.stack_id.hex}",
  "environment_name": "_default",
  "host_ip": "${vsphere_virtual_machine.NODENAME.network_interface.0.ipv4_address}",
  "node_name": "${var.NODENAME-name}",
  "runlist": "role[ROLE]",
  "node_attributes": {
    "ibm_internal": {
      "roles": "[ROLE]"
    }
  }
}
EOT
}
```

The following variables must be substituted:

- **NODENAME**: The logical name of the node within the Terraform template.
- **ROLE**: The name of the Chef role to be executed.

Template variables modification

Managed services Variables update guidance.

Managed services Variables provide the opportunity for an administrator to hardcode certain values for a parameter and make them relevant for a specific organization. Additionally, the deployment properties of variables may be modified to make suitable for a specific situation.

Managed services Variables may be accessed and edited from the **Templates** tab in the Template Editor.

Figure. Variables editing

When to customize

Managed services templates are designed to multi-cloud and multi-purpose environments. The correct way to use CAM Variables is to customize a template for a specific use case, for example:

- Specific cloud endpoint
- Specific Middleware configuration (Development/Test/Production)

A systems administrator can set in advance and hide deployment details from the deployer to achieve a specific effect during the deployment.

Hiding a Managed services Variable from the deployer

To hide a Managed services Variable from the deployer and therefore set the variable for a specific template, you must modify the following fields in CAM Variables. The **`default`** value is injected in the Terraform template at deploy time.

- **`hidden`**: Set this field to **`true`**.
- **`default`**: Set this value to be the default value.

Setting default value

Default values for Managed services Variables are a common use case and provide meaningful advice to the deployer on the correct value.

- **`default`**: Set this value to be the default value.

Field validation

Field validation occurs through a Regular Expression described in the **`validation`** parameter.

- **`validation`**: Set to the relevant Regular Expression.

Chef guidelines with Managed services

The following is a general advice and guidance on how to integrate Chef Cookbooks with Managed services and the essential structural elements of Chef as used within CAM.

Fundamentals

Role based orchestration

The most important aspect of integrating existing Chef cookbooks with Managed services is that all orchestration from the template is based upon the Chef Role. Chef Roles are the primary artifact for orchestration.

When you examine the Terraform Resource responsible for calling Chef through the Pattern Manager, you notice that the Chef Role is mentioned as the runlist. Subsequent calls to Chef appends the Role to the RunList in order.

```
resource "camc_softwaredeploy" "centralnode_liberty_install" {
  depends_on =
```

```
["camc_bootstrap.centralnode_chef_bootstrap_comp","camc_bootstrap.libertynode_chef_bootstrap_comp"]
  name = "centralnode_liberty_install"
  camc_endpoint = "${var.ibm_pm_service}/v1/software_deployment/chef"
  access_token = "${var.ibm_pm_access_token}"
  skip_ssl_verify = true
  trace = true
  data = <<EOT
{
  "os_admin_user": "${var.centralnode-os_admin_user}",
  "stack_id": "${random_id.stack_id.hex}",
  "environment_name": "_default",
  "host_ip": "${vsphere_virtual_machine.centralnode.network_interface.0.ipv4_address}",
  "node_name": "${var.centralnode-name}",
  "runlist": "role[liberty_install]",
  "node_attributes": {
    "ibm": {
      "im_repo": "${var.ibm_im_repo}",
      "im_repo_user": "${var.ibm_im_repo_user}",
      "sw_repo": "${var.ibm_sw_repo}",
      "sw_repo_user": "${var.ibm_sw_repo_user}"
    },
    "ibm_internal": {
      "roles": "[liberty_install]"
    },
    "was_liberty": {
      "base_version": "${var.centralnode_was_liberty_base_version}",
      "edition": "${var.centralnode_was_liberty_edition}",
      "im_install_dir": "${var.centralnode_was_liberty_im_install_dir}",
      "install_dir": "${var.centralnode_was_liberty_install_dir}",
      "install_grp": "${var.centralnode_was_liberty_install_grp}",
      "install_user": "${var.centralnode_was_liberty_install_user}",
      "java_version": "${var.centralnode_was_liberty_java_version}",
      "wlp_user_dir": "${var.centralnode_was_liberty_wlp_user_dir}"
    }
  },
  "vault_content": {
    "item": "secrets",
    "values": {
      "ibm": {
        "im_repo_password": "${var.ibm_im_repo_password}",
        "sw_repo_password": "${var.ibm_sw_repo_password}"
      }
    },
    "vault": "${random_id.stack_id.hex}"
  }
}
EOT
}
```

In this example, the **roles** field is populated with the role to be run. The *Pattern Manager* appends the role to the runlist of the node and runs the chef-client to complete the execution of the component.

Chef Cookbooks must therefore be designed in such a way as to be orchestrated by the running and subsequent running of Chef Roles. It is important to note the relationship between *Attributes* passed from Managed services and the Chef Role. As a general rule, hide all attributes that are not to be changed within a Role and expose only *Node* based attributes to Managed services.

Chef Client and Server version

When a *Content Runtime* is deployed, a pre-approved version of the Chef Client and Server are deployed. As Chef is an open source tool, only specific versions of the Chef Client and server are approved for deployment. The current versions are as follows:

- Chef Server: v12.11.1
- Chef Client: v12.12.13

Language linting

IBM provided Chef Cookbooks use the following tools to standardize on the language and structure of Chef Cookbooks:

-Rubocop -FoodCritic

Testing tooling

IBM tests all cookbooks with the following tools that are automated as part of a Devops Toolchain.

- ChefSpec

- Test Kitchen

README generation

All Chef Cookbook *README.md* files are generated with the knife readme plugin that is located at Chef.

Structural elements

The following section is a summary of the structural elements of standard IBM provided cookbooks. It is not necessary for customer provided cookbooks to follow these standards, they are included as a reference only.

Standard Chef attributes

Chef Node Attributes are commonly used to define cross-cookbook attributes, which may be used to describe common features. A single *Chef Node* might consume multiple cookbooks, so standardize those standard variables that affect all cookbooks.

The following is the list of standard Chef Attributes used in IBM provided cookbooks. These may be referenced in a customer provided cookbook only if there is a need to re-use these standard variables.

Note: This is not mandatory but required only for consistency and simplicity.

Attribute	Definition
default['ibm']['expand_area']	Home directory for Open Patterns unfold directory, this is a global variable that may be set in the Environment. This path is absolute and all other paths are relative to it.
force_override[ ]['expand_area']	Cookbook based ExpandArea which will be relative to the global expandArea. This value will be set in the internal.rb file.
default['ibm']['sw_repo_root']	URL location of the Repository Server.
force_override[' ']['archive_names']	The correct version attribute name to describe the archive to install a product from is [' '][' '][''v ']['archive_names']. This attribute may be represented by an array if necessary, each element having a reasonable descriptor of the function of the installer in question. The SHA Value must also be represented for each file.
force_override[' ']['fixpack_names']	ARRAY of file names of the installer.
force_override[ibm][evidence_path]['windows']	Root path for the Evidence files on Windows systems, this variable will be defined in a role on the customer chef server.
force_override[ibm][evidence_path]['unix']	Root path for the Evidence files on Unix systems, this variable will be defined in a role on the customer chef server.
default[ ]['evidence_zip']	ZIP File which contains the evidence.
default[' ']['fixpack_id']	Identifier for the fix pack to be applied. This attribute is necessary for all products that support the notion of applying fix packs. The precise fix pack must never be hard-coded in the Recipe.
force_override[' ']['os_users']	The data structure should include a hash for every operating system user including related settings such as username, home directory, password, ldap_user etc.
Base directory for a product to be installed into.	Home directory for Open Patterns unfold directory, this is a global variable that may be set in the Environment. This path is absolute and all other paths are relative to it.
force_override[' ']['sw_repo_path']	One or more attributes defining the relative location of product related files on the software repository (such as default['ibm']['sw_repo_root']).
force_override[' ']['fp_repo_path']	One or more attributes defining the relative location of fix pack related files on the software repository (such as default['ibm']['sw_repo_root']).
force_override[' ']['os_libraries']	The value should contain a list of operating system libraries to install.
default['ibm']['temp_dir']	An absolute path to a temporary location on the filesystem to store any temporary files.
default['ibm']['log_dir']	Any logfiles that should persist on the system should be written to this location.

Avoiding namespace clashes

As a node might run one or more cookbooks, seperate similar attributes from different cookbooks. The following rule applies when naming attributes.

```
['<cookbookName>']['<attribute>']
```

Internal.rb vs Default.rb

IBM categorizes Chef variables based on whether they are to be exposed in CAM or not, the general structure applies.

```
|- chef
|    |- cookbooks
|    |    |- <cookbook_name>
|    |- attributes
|         - default.rb              # All product attributes
|         - internal.rb             # force_override attributes that may never be changed.
```

Attribute Precedence

Chef supports a wide variety of attribute precedence, the following is a summary of the supported and the relevant use cases.

Standard Chef Recipes

The following is the set of standard recipes that should exist in all IBM provided cookbooks. Product specific cookbooks to configure and control are added separately.

Receipe Name	Definition
prereq	OS Pre-requisite provisioning.
prereq-check	Recipe to ensure that pre-requisites are in place for a cookbook to run.
fixpack	Product un-installation only. (Not necessary by default, however, should it be required, then use this name)
uninstall	OS Pre-requisite provisioning.
harden	Product hardening only. Note, this is inherently customer specific and must therefore be separated from other recipes.
install	Product installation only.
gather_evidence	Recipe to gather artifacts to prove an installation has occurred successfully. These artifacts are to be packed in a single archive.
cleanup	Recipe to remove all unwanted files such as install media and temp files.
default	Standard default recipe which will implement the minimal functionality to stand up the cookbook.

Chef Cookbook Principles

The following is a list of principles used in the construction of Chef Cookbooks.

The Granularity Principle

Granularity in the CHEF context refers to the care taken to expose recipes in a granular enough fashion to ensure Orchestration and Pattern designers can assemble the component pieces in such a way as to gain the desired result. There are no hard and fast rules here, the general principal to apply is this:

```
Recipes should seek to encapsulate a single software function.
```

Recipes represent the lowest representation of functionality provided by the Chef Cookbook and as such should contain with them a single operation that may be combined into a Role as a Pattern is defined based on these recipes. Standard CHEF Recipes represent both a minimal default and also encapsulate this principle.

Configuration of software products is in general difficult to generalize, however, consider a similar principle when deciding on Recipes that seek to implement software configuration. As a general rule, a single recipe should implement a single configuration concept.

When evaluating a Software Product to try to find the correct level of Granularity, it is important to be aware of the following general principles.

- How Recipes relate to Roles.
- How Roles are used in Pattern definition.
- How Roles are used in Pattern Orchestration.
- An understanding of the standard Recipes required for any software product.

  An understanding of where Recipes sit in relation to the overall Open Pattern construct is essential to an understanding of the best way to find the correct level of granularity.

The Idempotency Principle

The Idempotency principal is in essence the practice of running an action more than once producing the same result. From a CHEF perspective, assume that a recipe will fail at some point and needs to be re-run.

```
All CHEF Cookbook's and Recipe's must be inherently idempotent.
```

Recipes must be designed with re-entry in mind. This can be as a result of failure, or as a result of the chef client being run at unexpected times. Cookbooks should make no assumption's as to the start state of the system and always check that an action can be run before it is run.

Touch points include:

- Only download files where necessary, for example, do not download installer files if the installation is already complete.
- Only execute configuration changes.
- All LWRP and other CHEF artefacts for example Recipes must be inherently idempotent.
- Cookbooks should never produce errors based on a state of the running software.

Data Driven Principle

Data Driven refers to the exposing as many options as CHEF variables as possible and therefore reducing the number of hard coded values. The general principle is as follows: CHEF Cookbooks should be highly dependent on attributes when executing, changing the outcome should be driven by modifying the attributes through the use of Roles and Node Attributes. The following generalizations apply:

- Maximize CHEF Attributes.
- Minimize hard coding.
- CHEF Attributes should be represented as arrays where possible.

Maximizing Attributes / Minimize Hard Coding As a general rule, values that are consumed by a cookbook should be represented as Attributes. Attribute definition and consumption by CHEF should follow the guidelines set out in the Attribute Precedence section of this document. As a general Rule all attributes are changeable, however, there are circumstances where certain variables are to remain fixed, these are given the force_override operator and defined in the internal.rb file.

Standard Attributes Attributes that implement standardized concepts should be named as per the Naming Standards described in this guide. Standard Attributes are defined to maximize readability of our Chef Cookbooks, but also to allow for attributes to be standardized across an environment.

Attribute Arrays Software products that implement multiple configuration items of the same type should be represented in an Array from a definition perspective. A reasonable example of this is the definition of database, which may be described in terms of 1..n databases. It is best practice in these cases to define the attributes in an array rather than singleton attributes, for example:

```
default['was']['os_users'] = {
  'wasadmin'  =>  {
      'name' =>     'wasadmin',
      'gid' =>      'wasgrp',
      'comment' =>  'WAS administrative user',
      'home' =>     "/home/wasadmin",
      'shell' =>    '/bin/bash'
  },
  'wasowner'  =>  {
      'name' =>     'wasowner',
      'gid' =>      'wasgrp',
      'comment' =>  'WAS owner',
      'home' =>     "/home/wasowner",
      'shell' =>    '/bin/bash'
  }
}
```

The previous example defines the os_users attribute as an Hash Table rather than as singleton attributes. It is therefore possible to add or remove users based on data alone, without the need to define any new attributes or change the code in general.

Modularity Principle

The principle of Modularity relates to creating re-usable components that may be utilized within a single cookbook, or to external cookbooks.

`CHEF Cookbooks should be modular in nature and take advantage where possible of Libraries, LWRP and other CHEF Cookbooks.`

An important consideration when designing a cookbook is whether to use an LWRP or not. A Chef Lightweight Resource Provider can be created to encapsulate the functionality of Software Configuration items to be treated as an object. A clear identifier of an object that should be represented as an LWRP is that it may be modified, exhibits internal state, applies across Operating Systems and is in general version independent. As a guide, consider existing CHEF Resources, each of these are in fact an implementation of a LWRP.

Never attempt to re-create an existing Chef Resource if one already exists. Open Patterns should maximize the use of LWRP for the following situations.

- Fix pack installation.
- State control (For example, stop/start) Software configuration.

LWRP should exhibit the following characteristics.

- Operating System neutral.
- Software version neutral.
- State neutral. (For example, start a Software Product that is already started will not cause an error)
- Idempotent.

Modifying a deployed middleware template instance

Managed services middleware template modification

You can redeploy an existing instance after selecting the desired template version and modifying the input parameters. Instructions for doing so can be found on the modifying a deployed template instance. There are a few restrictions with modifying the middleware templates. Those restrictions are listed here.

Chef assets cannot be updated

The Chef cookbooks in the Managed services (CAM) middleware templates are not written to support modification after deploy. This fits with the cloud model of quickly destroying and recreating entities as needed. When attempting to modify a deployed middleware instance, you will see the node properties are not able to be edited.

The properties are not editable due to a property in the template's CAM variables file called immutable_after_create. If you are writing your own templates and would like to prevent properties from being modified after deploy, then the immutable_after_create property is what you would use. Similarly, if you have cloned the middleware templates to your own git repository and would like to be able to modify these parameters, you would update the immutable_after_create property in the template's CAM variables file.

For more information on Managed services variables, see Template variables modification

Terraform Provider Configuration

Starting from Terraform 0.13.0 a new hierarchical namespace for providers was introduced by Terraform. For more information on this refer to Terraform documentation. This new feature requires explicit source information for any providers that are not maintained by Hashicorp. Terraform 0.13.0 and above provides a new nested block **`required_providers`** inside the **`terraform`** configuration block to specify the provider source location. The value for **`source`** attribute is in the format **`<REGISTRY_HOST>/<NAMESPACE>/<PROVIDER_TYPE>`**

The starterlibrary provides the following templates that showcase how to use this new source syntax.

- Google Cloud Single Virtual Machine Example for Terraform 1.x.x This template shows how to source the provider maintained by hashicorp and hosted in default hashicorp registry. The value **`hashcorp/google`** assigned to attribute **`source`** refers to the location of provider binary which is packaged by default under **`$HOME/.terraform.d/plugins/registry.terraform.io/hashicorp/google`** in the **`cam-provider-terraform`** pod. The name **`hashicorp`** in the source value indicates that the provider is maintained by Hashicorp. The value **`registry.terraform.io`** in the binary path refers to the default provider registry host which can be ommitted in the source value as the provider is in the default registy.
- Virtual Server in a new VPC with SSH key For Terraform Engine 13 and 14 This template shows how to source the provider maintained by a third party vendor which in this case is IBM and hosted in default hashicorp registry. The value **`IBM-Cloud/ibm`** assigned to attribute **`source`** refers to the location of provider binary which is packaged by default under

$HOME/.terraform.d/plugins/registry.terraform.io/IBM-Cloud/ibm in the **cam-provider-terraform** pod. The name **IBM-Cloud** in the source value indicates that the provider is a thrid-party provider. The value **registry.terraform.io** in the binary path refers to the default provider registry host which can be ommitted in the source value as the provider is in the default registry.

For the templates that use the IBM Cloud Pak for AIOps Managed services provided Terraform providers, you must use the **required_providers** section as shown in the following sample.

```
terraform {
  required_version = ">= 0.13"

  required_providers {
    camc = {
      #Source attribute value is in format <REGISTRY_HOST>/<NAMESPACE>/<PROVIDER_TYPE>
      source  = "registry.ibm.com/cam/camc"
      version = "~> 0.2"
    }
  }

}
```

The above sample defines the source location for IBM Cloud Pak for AIOps Managed services provided CAMC Provider. The value **registry.ibm.com/cam/camc** assigned to attribute **source** refers to the location of provider binary which is packaged under **$HOME/.terraform.d/plugins/registry.ibm.com/cam/camc** in the **cam-provider-terraform** pod. The name **registry.ibm.com/cam** in the source value indicates that the provider is a IBM Cloud Pak for AIOps Managed services provided provider. The value **registry.ibm.com** in the binary path refers to a local dummy non-existent domain to indicate that this provider is a local in-house provider.

For a different IBM Cloud Pak for AIOps Managed services provided provider say UCD provider, you would change only the value for **PROVIDER_TYPE** as shown below.

```
terraform {
  required_version = ">= 0.13"

  required_providers {
    ucd = {
      #Source attribute value is in format <REGISTRY_HOST>/<NAMESPACE>/<PROVIDER_TYPE>
      source  = "registry.ibm.com/cam/ucd"
      version = "~> 0.5"
    }
  }

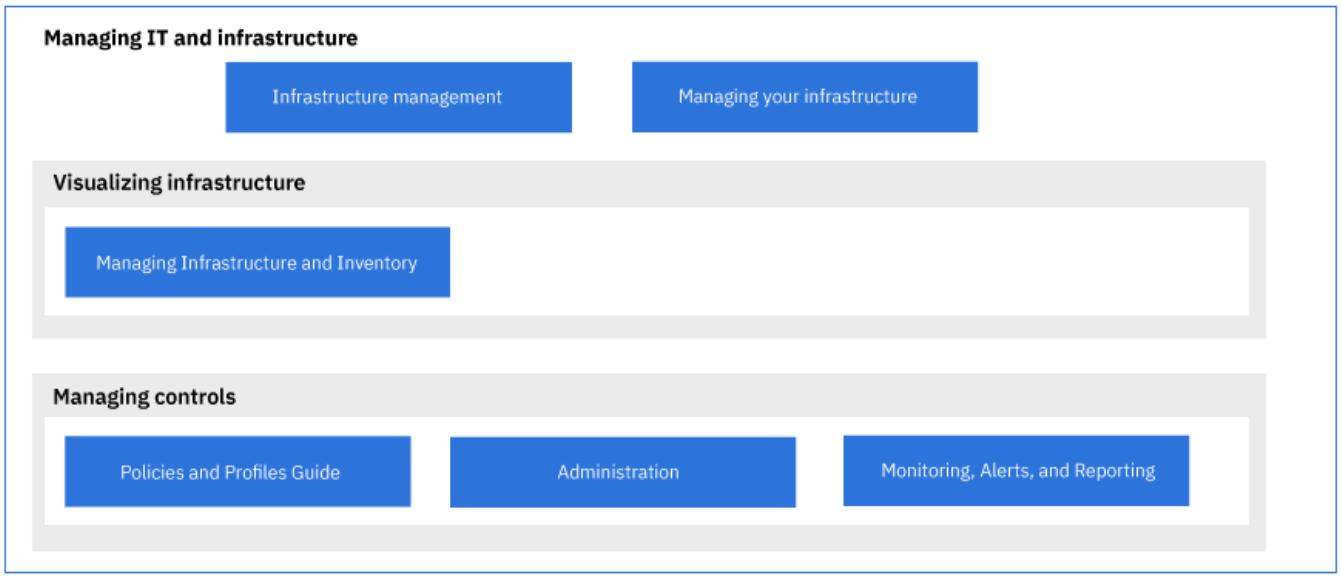
}
```

Refer to Shipped Terraform providers for complete set of provider binaries (which are compatiable with Terraform 0.13.0 or higher) bundled with IBM Cloud Pak for AIOps Managed services.

Managing IT and infrastructure

Learn more about managing IT and infrastructure with Infrastructure Automation.

To learn more about managing IT for Infrastructure Automation, click a subject within the following image to view the related available documentation (or click the subject from list that follows the image):



- Infrastructure management
- Managing your infrastructure
- Visualizing infrastructure
  - Managing Infrastructure and Inventory
- Managing controls
  - Policies and Profiles Guide
  - Administration
  - Monitoring, Alerts, and Reporting

Infrastructure Management

Infrastructure Automation – Infrastructure Management delivers the insight, control, and automation enterprises need to address the challenges of managing virtual environments, which are far more complex than physical ones. This technology enables enterprises with existing virtual infrastructures to improve visibility and control, and those just starting virtualization deployments to build and operate a well-managed virtual infrastructure.

Infrastructure management includes the following feature sets:

- Insight: discovery, monitoring, utilization, performance, reporting, analytics, chargeback, and trending.
- Control: security, compliance, alerting, policy-based resource and configuration management.
- Automate: IT process, task and event, provisioning, workload management and orchestration.
- Integrate: systems management, tools and processes, event consoles, CMDB, RBA, and web services.

Note: When the Infrastructure management component of Infrastructure Automation is installed, you can access the Infrastructure management UI from the IBM Cloud Pak Automation console UI by browsing to the menu: **Automate Infrastructure > Infrastructure management**. The documentation for Infrastructure management, including the menu navigation for documented procedures, is written from the context of first browsing to the Infrastructure management UI.

This documentation is delivered by various sources in the ManageIQ open source community. IBM accepts support tickets, and corrects documentation errors. Support and enhancement updates can also come from the open source community.

- Appliance Installation and Upgrade
- Managing your infrastructure
- Administration
- Reference
- Integration

Infrastructure Management

Viewing and collecting information from your clusters, hosts, virtual machines, and other resources.

- Policies and Profiles Guide
- Managing Infrastructure and Inventory
- Managing Providers
- Creating a Service in Infrastructure Management for Virtual Machine Provisioning
- Provisioning Virtual Machines and Instances
- Service Provisioning in Infrastructure Management using an orchestration template

Visualizing infrastructure

Learn more about visualizing infrastructure with Infrastructure Automation.

- Managing Infrastructure and Inventory

- Managing Infrastructure and Inventory
- Clusters
- Performing SmartState Analysis on Clusters
  - Comparing Clusters
- Creating a Cluster Comparison Report
- Viewing a Cluster
- Tagging Clusters
- Viewing Capacity and Utilization Charts for a Cluster
- Viewing Cluster Timeline
- Detecting Drift on Clusters
- Creating a Drift Report for Clusters
- Removing Clusters
- Hosts
  - Filtering Hosts
- Setting a Default Host Filter
- Creating a Host Filter
- Performing SmartState Analysis on Hosts
  - Comparing Hosts
- Host Comparison Sections
- Using the Host Comparison Sections
- Creating a Host Comparison Report
- Refreshing Multiple Hosts
- Adding a Single Host
- Editing Hosts
- Viewing a Host
- Tagging Multiple Hosts
- Removing Hosts
  - Scaling Down Compute Hosts
- Refreshing Relationships and Power States for a Host
- Viewing Capacity and Utilization Charts for a Host
- Viewing the Host Timeline
- Host Virtual Summary
- Viewing Host Device Information
- Viewing Host Network Information
- Viewing Storage Adapters
- Detecting Drift on Hosts
- Creating a Drift Report for Hosts
- Virtual Machines
  - Filtering Virtual Machines and Templates
- Creating a Virtual Machine or Template Filter
- Loading a Report Filter or Search Expression
- Changing Views for Virtual Machines and Templates
- Sorting Virtual Machines and Templates
- Creating a Virtual Machine or Template Report
- Searching for Virtual Machines or Templates
  - Analyzing Virtual Machines and Templates

    - Red Hat Enterprise Virtualization Prerequisites
- SmartState Analysis on Red Hat Enterprise Virtualization Manager 3.1 and Above - Storage Support Notes
- SmartState Analysis on Red Hat Enterprise Virtualization Manager 3.0 - Storage Support Notes
    - VMware vSphere Prerequisites
    - Comparing Virtual Machines and Templates
- Virtual Machine and Templates Comparison Sections
- Using the Virtual Machine Comparison Sections
- Creating a Virtual Machine Comparison Report
- Controlling the Power State of Red Hat Virtualization Virtual Machines
- Refreshing Virtual Machines and Templates
- Extracting Running Processes from Virtual Machines and Templates
- Setting Ownership for Virtual Machines and Templates
- Removing Virtual Machines and Templates from the VMDB
- Tagging Virtual Machines and Templates
- Viewing a VMware Virtual Machine's Storage Profile
- Viewing Running Processes after Collection
- Editing Virtual Machine or Template Properties
- Setting Ownership of a Virtual Machine or Template
- Right Sizing a Virtual Machine
- Viewing Capacity and Utilization Charts for a Virtual Machine
- Viewing the Virtual Machine or Template Timeline
- Virtual Machine or Template Summary
- Viewing the Operating System Properties
- Viewing Virtual Machine or Template Snapshot Information
- Viewing User Information for a Virtual Machine or Template
- Viewing Group Information for a Virtual Machine or Template
- Viewing Genealogy of a Virtual Machine or Template
- Comparing Genealogy of a Virtual Machine or Template
- Tagging Virtual Machines or Templates with a Common Genealogy
- Detecting Drift on Virtual Machines or Templates
- Creating a Drift Report for a Virtual Machine or Template
- Viewing Analysis History for a Virtual Machine or Template
- Viewing Disk Information for a Virtual Machine or Template
- Reconfiguring Virtual Machines
- Adding a Disk to a Virtual Machine
- Removing a Virtual Machine Disk
- Resizing a Virtual Machine Disk
- Increasing or Decreasing a Virtual Machine's Memory Size
- Reconfigure a Virtual Machine's Processors
- Adding or Removing Virtual Machine Network Adapters
    - Adding a Network Adapter to a VMware Virtual Machine
    - Removing a Network Adapter from a VMware Virtual Machine
- Attach or Remove an ISO (VMware Virtual Machines Only)
- Viewing Event Logs for a Virtual Machine or Template
- Remote Consoles
    - Configuring Console Access to VMware ESXi Hosts At A Network Layer
        - Using SSH to Configure VMware ESXi Hosts to Enable Console Access
        - Using the VMware vCenter Web Interface to Configure ESXi Hosts to Enable Console Access
        - Configuring the VMware ESXi Host Firewall Ports for Console Access
    - Configuring WebMKS Support in Infrastructure Automation
    - Opening a Console for a Virtual Machine
- Managing Virtual Machine Snapshots
- Creating a Virtual Machine Snapshot
- Deleting a Virtual Machine Snapshot
- Reverting a Virtual Machine to a Snapshot
- Creating a Template Based on a Virtual Machine
- Retiring Virtual Machines
- Migrating Virtual Machines
- Resource Pools
- Removing a Resource Pool
- Tagging a Resource Pool

- Viewing the Resource Pool Summary
- Resource Pools Accordion
- Datastores
- Performing SmartState Analysis on Datastores
- Viewing a Datastore
- Tagging a Datastore
- Viewing Capacity and Utilization Charts for a Datastore
- Removing a Datastore
- Data Optimization
  - Utilization Trends
    - Viewing a Utilization Trend Summary
    - Viewing Detail Lines of a Utilization Trend
    - Viewing a Report of a Utilization Trend
- PXE Servers
- Availability Zones
- Viewing an Availability Zone
- Viewing Availability Zone Relationships
- Cloud Tenants
  - Tenant Mapping
- Viewing Cloud Tenant Relationships
- Volumes
  - Amazon Elastic Block Store Manager Volumes
- Creating Volumes
- Creating a Snapshot of a Volume
- Attaching a Volume to an Instance
- Detaching a Volume from an Instance
- Editing a Volume
- Deleting a Volume
  - OpenStack Block Storage Manager Volumes
- Creating Volumes
- Creating a Backup of a Volume
- Restoring a Volume from a Backup
- Restoring a Cloud Volume from a Backup
- Deleting a Cloud Volume Backup
- Creating a Snapshot of a Volume
- Attaching a Volume to an Instance
- Detaching a Volume from an Instance
- Editing a Volume
- Deleting a Volume
  - IBM Power Systems Virtual Servers Block Storage Manager Volumes
- Creating Volumes
- Attaching a Volume to an Instance
- Detaching a Volume from an Instance
- Deleting a Volume
- Flavors
- Creating a Flavor
- Viewing a Flavor
- Viewing Flavor Relationships
- Deleting a Flavor
- Cloud Networks
  - Creating and Administering Cloud Networks
    - Adding a Cloud Network
    - Editing Cloud Network Details
    - Deleting a Cloud Network
  - Creating and Administering Subnets
    - Adding a Subnet
    - Editing a Cloud Subnet
  - Configuring Network Routers
    - Adding a Network Router
    - Editing Network Router Details
    - Adding an Interface to a Network Router
    - Removing a Network Router Interface

        - Deleting a Network Router
    - Creating Floating IPs
        - Adding Floating IPs.
        - Managing Port Association of a Floating IP
        - Deleting a Floating IP
    - Security Groups
        - Editing Security Group Details
        - Viewing Security Groups
        - Tagging Security Groups
        - Deleting a Security Group
- Instances and Images
    - Filtering Instances and Images
- Using an Instance or Image Filter
- Creating an Instance or Image Filter
- Loading a Report Filter or Search Expression
- Changing Views for Instances and Images
- Sorting Instances and Images
- Creating an Instance or Image Report
- Searching for Instances or Images
- Analyzing Instances and Images with SmartState Analysis
    - Restrictions on Displaying Files Collected
    - Comparing Instances and Images
- Creating an Instance Comparison Report
- Refreshing Instances and Images
- Extracting Running Processes from Instances and Images
- Setting Ownership for Instances and Images
- Removing Instances and Images from the VMDB
- Tagging Instances and Images
- Reviewing an Instance or Image
- Viewing Running Processes after Collection
- Managing Security Groups for Cloud Instances
- Editing Instance or Image Properties
- Controlling the Power State of an Instance
- Right Sizing an Instance
    - Resizing an Instance
    - Migrating a Live Instance
    - Evacuating an Instance
- Viewing Capacity and Utilization Charts for an Instance
- Viewing the Instance or Image Timeline
- Viewing the Instance or Image Summary
- Viewing User Information for an Instance or Image
- Viewing Group Information for an Instance or Image
- Viewing Genealogy of an Instance or Image
- Detecting Drift on Instances or Images
- Creating a Drift Report for an Instance or Image
- Viewing Analysis History for an Instance or Image
- Viewing Event Logs for an Instance or Image
- Orchestration Stacks
- Tagging Orchestration Stacks
- Retiring Orchestration Stacks
- Removing Orchestration Stacks
- Key Pairs
- Adding a New Key Pair
- Removing a Key Pair
- Downloading Key Pairs
- Object Stores
- Tagging Object Stores
- VMware Networking Switches
- Tagging VMware Networking Switches
- Container Entities
    - The Containers Overview Page
    - Viewing a Container Entity Summary

- Using the Topology Widget
  - Viewing the Topology for Container Providers
  - Viewing the Topology for Container Provider Projects
  - Limiting the Number of Containers Shown in the Topology View
- Analyzing Container Images with SmartState Analysis
- Configuring Automatic Tagging for Container Entities

Managing Infrastructure and Inventory

Clusters

Clusters provide high availability and load balancing for a group of hosts. The **Clusters** page under **Compute > Infrastructure** displays the clusters discovered in your enterprise environment.

Note:

Any filter applied will be in effect here.

Use the **Clusters Taskbar** to manage the analysis and tagging of your clusters. These buttons manage multiple clusters at one time. To manage one cluster, click on that cluster in the main area of the screen.

Performing SmartState Analysis on Clusters

Analyze a cluster to gather historical data to compare with previous points in time.

1. Browse to menu: **Compute > Infrastructure > Clusters**.
2. Check the clusters to analyze.
3. Click **Configuration**, and then (**Perform SmartState Analysis**).
4. Click **OK**.

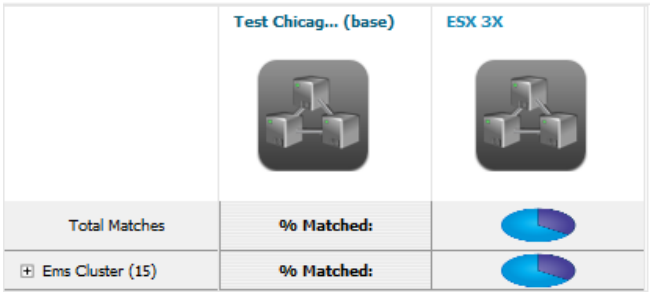
The SmartState Analysis begins and returns the current data.

Comparing Clusters

Infrastructure Automation provides features to compare properties of clusters.

1. Browse to menu: **Compute > Infrastructure > Clusters**.
2. Check the clusters to compare.

3. Click **Configuration**, and then (**Compare Selected items**). The comparison displays in a default expanded view and lists a limited set of properties.



4. To delete a cluster from the comparison, click (**Remove this Cluster from the Comparison**).
5. To go to a compressed view, click (**Compressed View**). To return to an expanded view, click (**Expanded View**).
6. To change the base cluster that all other clusters compare to, click its label at the top of its column.
7. To go to the cluster summary screen, click its virtual thumbnail or icon.
8. There are three buttons in the taskbar to limit the type of views:
   - Click (**All attributes**) to see all attributes.
   - Click (**Attributes with different values**) to see only the attributes that are different across clusters.
   - Click (**Attributes with the same values**) to see only the attributes that are the same across clusters.
9. To limit the mode of the view, there are two taskbar buttons.
   - Click (**Details Mode**) to see all details for an attribute.
   - Click (**Exists Mode**) to only see if an attribute exists compared to the base or not. This only applies to attributes that can have a Boolean property. For example, a user account exists or does not exist, or a piece of hardware that does or does not exist.

This creates a comparison between clusters. Export this data or create a report from your comparison for analysis using external tools.

Creating a Cluster Comparison Report

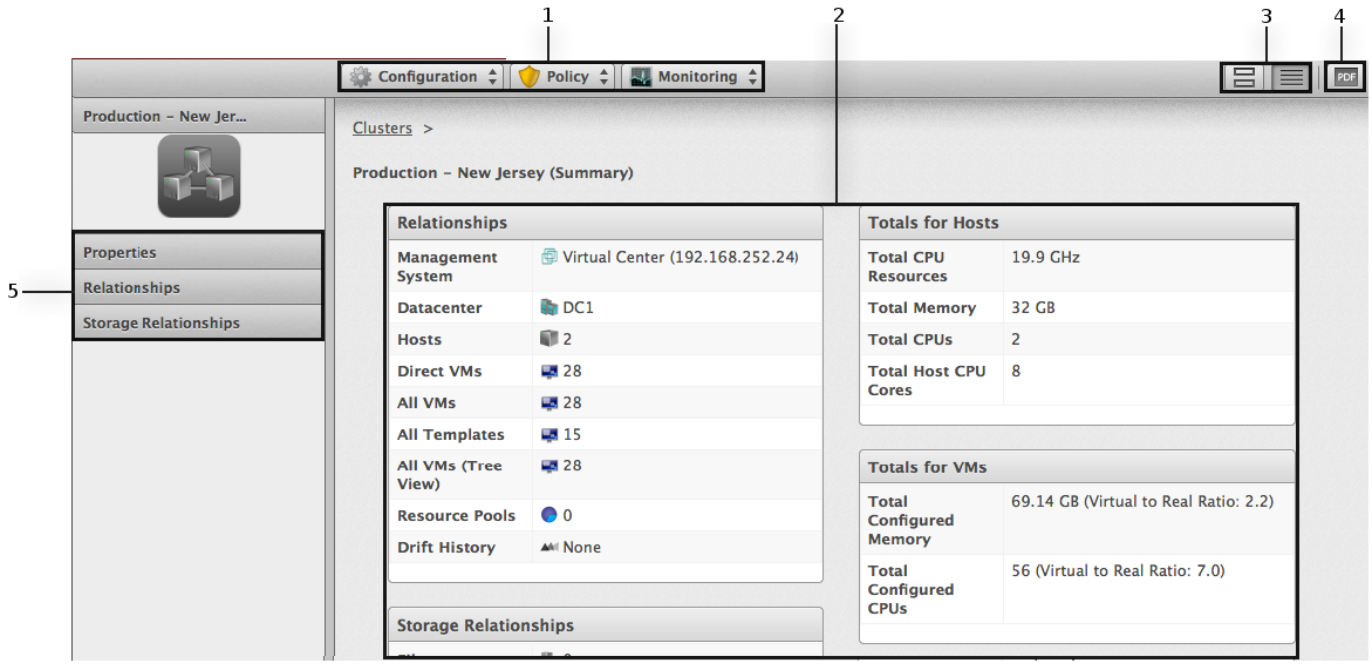
Create a quick report of to compare clusters in CSV, TXT, or PDF formats.

1. Create the comparison to analyze.
2. Click (**Download**).
3. Click the output button for the type of report.
   - Click (**Download comparison report in TXT format**) for a text file.
   - Click (**Download comparison report in CSV format**) for a comma-separated file.
   - Click (**Download comparison report in PDF format**) for a PDF file.

Viewing a Cluster

You can click on a specific cluster to view its details. The screen provides you with a cluster taskbar, a cluster accordion, and a cluster summary.

Cluster Management Screen.

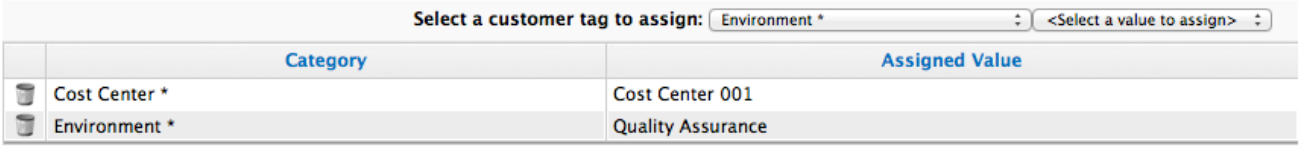


1. **Cluster Taskbar**: Choose between **Configuration**, **Policy** and **Monitoring** options for the selected cluster
2. **Cluster Summary**: See cluster summary such as **Relationships**, **Totals for Hosts**, **Totals for VMs**
3. **Cluster Summary Views**: Choose between graphical or text view of the cluster summary
4. **Cluster Summary PDF**: Generates cluster summary in PDF format
5. **Cluster Accordion**: See details about **Properties**, **Relationships**, **Storage Relationships** for the selected cluster

Tagging Clusters

Use tags to categorize clusters.

1. Browse to menu: **Compute > Infrastructure > Clusters**.
2. Check the Clusters to tag.
3. Click **Policy**, and then **Edit Tags**.

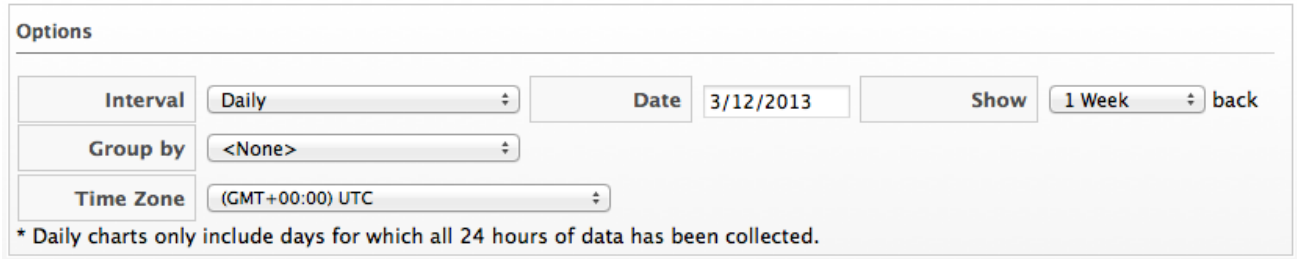


4. Select a customer tag from the first dropdown, and then a value for the tag.
5. Select more tags or click **Save** to save your changes.

Viewing Capacity and Utilization Charts for a Cluster

View capacity and utilization for a cluster.

1. Browse to menu: **Compute > Infrastructure > Clusters**.
2. Click the cluster to view Capacity and Utilization data.
3. Click (**Monitoring**), and then (**Utilization**) or from the accordion menu, click **Properties > Capacity & Utilization**.



4. From **Interval**, select to view hourly or daily data points and the dates to view data. Use **Group by** to group the lines by SmartTags. Use **Time Profiles** to select a time range for the data.

The **Capacity & Utilization** charts display

Note:

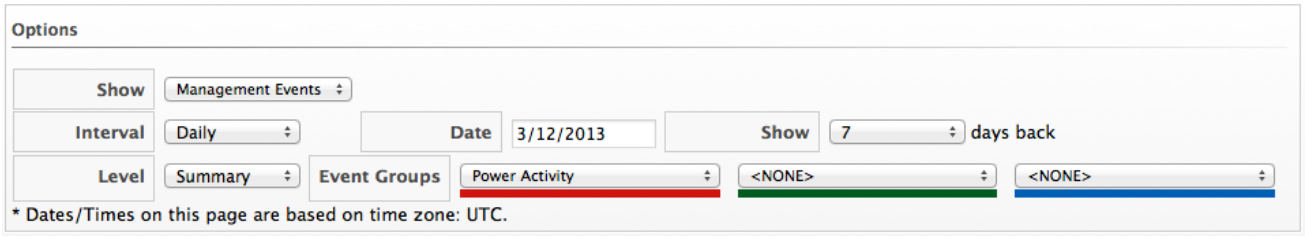
Daily charts only include full days of data. If a day does not include all the 24 data points for a day, the data does not show for that day.

For information about data optimization including utilization trend reports, see Data Optimization.

Viewing Cluster Timeline

Use the cluster timeline to see a graphical depiction of operational and configuration events over time.

1. Browse to menu: **Compute > Infrastructure > Clusters**.
2. Click the cluster to view the timeline.
3. Click (**Monitoring**), and then (**Timelines**) or from the accordion menu, click menu: **Properties > Timelines**.
4. From **Options**, customize the period of time to display, and the types of events to see.



- Use the **Interval** dropdown to select hourly or daily data points.
- Use **Date** to type the date for the timeline to display.
- If you select to view a daily timeline, use **Show** to set how many days back to go. The maximum history is 31 days.
- The three **Event Group** dropdowns allow the selection of different groups of events to display. Each has its own color.
- From the **Level** dropdown, select a **Summary** event if needed, or a **Detail** list of events. For example, the detail level of a *Power On* event might include the power on request, the starting event, and the actual **Power On** event. If you select **Summary**, the timeline only displays the Power On event.

5. To see more detail on an item in the timeline, click on it. A balloon appears with a clickable link to the resource.

Detecting Drift on Clusters

Over time, a cluster's configuration might change. Drift is the comparison of a cluster to itself at different points in time. The cluster requires analysis at least twice to collect information. Detecting drift provides users with the following benefits:

- See the difference between the last known state of a cluster and its current state
- Review the configuration changes that happen to a particular cluster between multiple points in time.
- Capture the configuration drifts for a single cluster across a time period.

Detect drift on clusters:

1. Browse to menu: **Compute > Infrastructure > Clusters**.
2. Click on the cluster to view drift.
3. Click **Relationships** in the cluster accordion.
4. Click **Drift History**.
5. Check the analyses to compare.
6. Click (**Drift Analysis**) at the top of the screen. The results are displayed.
7. Check the **Comparison** sections on the left to view in your comparison.
8. Click the plus sign next to the section name to expand it.
   - An item displayed on red text shows a change from the base analysis. An item displayed in black text shows no change from the base analysis.
   - A (**Changed from previous**) shows there has been a change since the last analysis.
   - A (**Same as previous**) means there has been no change since the last analysis.
   - Click (**Remove from drift**) at the bottom of a column to remove a specific analysis. The drift is then recalculated and the new results display.
9. Click (**Expanded View**) to see the expanded view. Click (**Compressed View**)] to compress the information.
10. Click the minus sign next to the section name to collapse it.
11. To limit the type of views, there are three buttons in the taskbar.

- Click (**All attributes**) to see all attributes of the sections selected.
- Click (**Attributes with different values**) to see only the attributes different across drifts.
- Click (**Attributes with the same values**) to see only the attributes the same across drifts.

The drift displays for your cluster. Download the data or create a report from the drift for analysis using external tools.

Creating a Drift Report for Clusters

Use the drift report feature to export information about your cluster's drift.

1. Create a drift of a cluster.
2. Click (**Download**).
3. Click the output button for the type of report you want.
   - Click (**Download drift report in TXT format**) for a text file.
   - Click (**Download drift report in CSV format**) for a comma-separated file.
   - Click (**Download drift report in PDF format**) for a PDF file.

Removing Clusters

If a cluster has been decommissioned or requires troubleshooting, it might require removal from the VMDB.

1. Browse to menu: **Compute > Infrastructure > Clusters**.
2. Check the clusters to remove.
3. Click **Configuration**, and then (**Remove Clusters from the VMDB**).
4. Click **OK**.

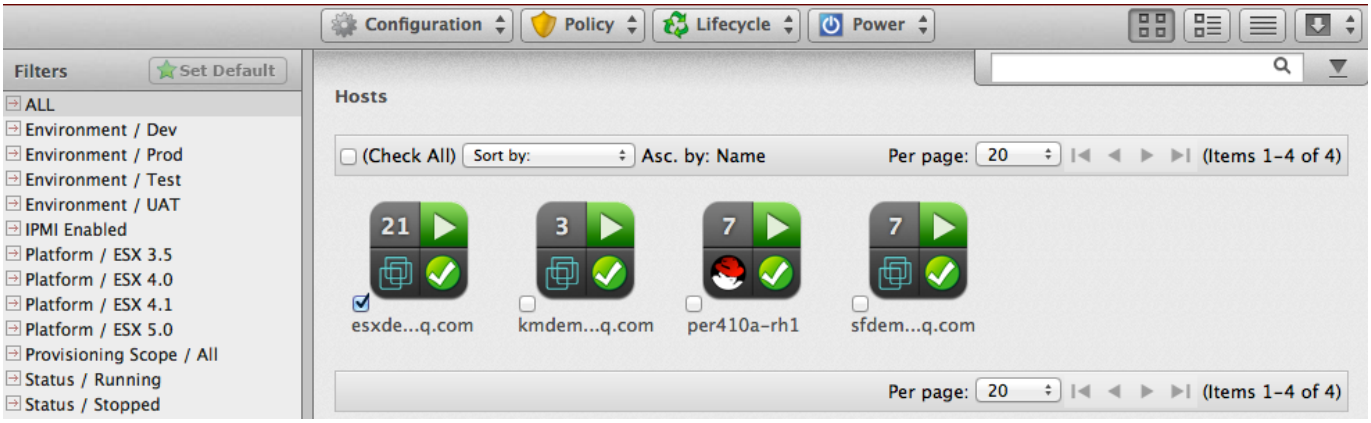
The clusters are deleted. Any virtual machines or hosts associated with these clusters remain, but are no longer associated with them.

Hosts

The **Hosts** page under **Compute > Infrastructure** displays the hosts discovered in your enterprise environment.

Note:

Any applied filters will be in effect here.



After adding or sorting your hosts, click on one to examine it more closely and see its virtual machines, SmartProxy settings, and properties.

1. Top left quadrant: Number of virtual machines on this host
2. Bottom left quadrant: Virtual machine software
3. Top right quadrant: Power state of host
4. Bottom right quadrant: Authentication status

Icon	Description
	Validated: Valid authentication credentials have been added.
	Invalid: Authentication credentials are invalid
	Unknown: Authentication status is unknown or no credentials have been entered.

Filtering Hosts

The Host Filter accordion is provided to easily navigate through the hosts. Use the ones provided or create your own. In addition, you can set a default filter.

Setting a Default Host Filter

Set the default filter for viewing your hosts.

1. From the **Filters** accordion on the left, click on the filter to use.
2. Click **Set Default** at the top of the filters list.

The default filter is set and marked by a green star next to its name.

Creating a Host Filter

Create a filter for viewing your hosts.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click (**Advanced Search**) to open the expression editor.
3. Use the expression editor to choose the appropriate options for your criteria.
4. Click **Save**.
5. Type in a name for the search expression in **Save this search as**.

   Note:

   This title depends on the type of resource you are searching.
6. Click **Save**.

The filter is saved and displays in the **My Filters** area of the **Filter** accordion.

Performing SmartState Analysis on Hosts

Perform a SmartState analysis on a host to collect additional information about it, such as patches, CPU, and memory.

Note:

- SmartState analysis on hosts is processed by the **Provider Operations** role. It is enabled by default.
- For ESX or ESXi hypervisors, consider the following: ESX hosts utilize a service console for host management and can be accessed using SSH. ESXi hosts lack a service console and therefore SSH cannot be used to obtain information sets for patches, services, Linux packages, user groups, SSH Config, and FS Files.
- `root` or administrator credentials are required to get patch information.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Check the hosts to analyze.
3. Click **Configuration**, and then (**Perform SmartState Analysis**).
4. Click **OK**.

Comparing Hosts

Infrastructure Automation allows you to compare hosts and check operating systems, host software and version information, and hardware.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Check the hosts to compare.
3. Click **Configuration**, and then (**Compare selected Hosts**). The comparison displays in a default expanded view, which lists a limited set of properties.
4. To remove a host from the comparison, click (**Remove this Host from the comparison**) at the bottom of the column.
5. To go to a compressed view, click (**Compressed View**). To return to an expanded view, click (**Expanded View**).
6. To limit the mode of the view, there are two buttons in the taskbar.
   - Click (**Details Mode**) to see all details for an attribute.
   - Click (**Exists Mode**) to limit the view to if an attribute exists compared to the base or not. This only applies to attributes that can have a Boolean property. For example, a user account exists or does not exist, or a piece of hardware that does or does not exist.
7. To change the base host that compare to the other hosts, click its label at the top of its column.
8. To go to the summary screen for a host, click its virtual thumbnail or icon.

Host Comparison Sections

Section	Description
Host Properties	Use this section to see basic information of the host, such as hostname, product, build number, hardware, and network adapters.
Security	Use this to see users and groups for the host, and firewall rules.
Configuration	Use this to see the operating system, applications, services, patches, vSwitches, vLANS, and advanced settings.
My Company Tags	Use this to see all tags.

Using the Host Comparison Sections

The following procedure describes how to use the host comparison sections.

1. On the left of a comparison screen, select the categories of properties to display.
2. Click the plus sign next to the sections name to expand it.

3. The following descriptions pertain to the **Expanded View** . Either the value of a property or an icon representing the property displays depending on the properties type.
    - A property displayed in the same color as the base means that the compared host matches the base for that property.
    - A property displayed in a different color from the base means that the compared host does not match the base for that property.
4. If you are in the **Compressed View** , the values of the properties do not display. All items are described by the icons shown below.
    - A (**checkmark**) means the compared host matches the base for that property. Hover over it and the value of the property displays.
    - A (**x**) means the compared host does not match the base for that property. Hover over it and the value of the property displays.
5. Click the plus sign next to the section name to collapse it.

This comparison is viewable in multiple ways. Export the data or create a report from your comparison for analysis using external tools.

Creating a Host Comparison Report

Create a quick report to compare clusters in CSV, TXT, or PDF formats.

1. Create the comparison to analyze.
2. Click (**Download**).
3. Click the output button for the type of report.
    - Click (**Download comparison report in TXT format**) for a text file.
    - Click (**Download comparison report in CSV format**) for a comma-separated file.
    - Click (**Download comparison report in PDF format**) for a PDF file.

Refreshing Multiple Hosts

Manually refresh a host for its properties and related infrastructure components.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Check the hosts to refresh.
3. Click **Configuration**, and then (**Refresh Relationships and Power States**).
4. Click **OK**.

When a host is refreshed and a new virtual machine is discovered on that host, Infrastructure Automation checks to see if the virtual machine is already registered with another host. If this is the case, the host that the virtual machine is associated with switches to the new host. If the SmartProxy is monitoring a provider, this happens automatically. If not, the next refresh of the host addresses this.

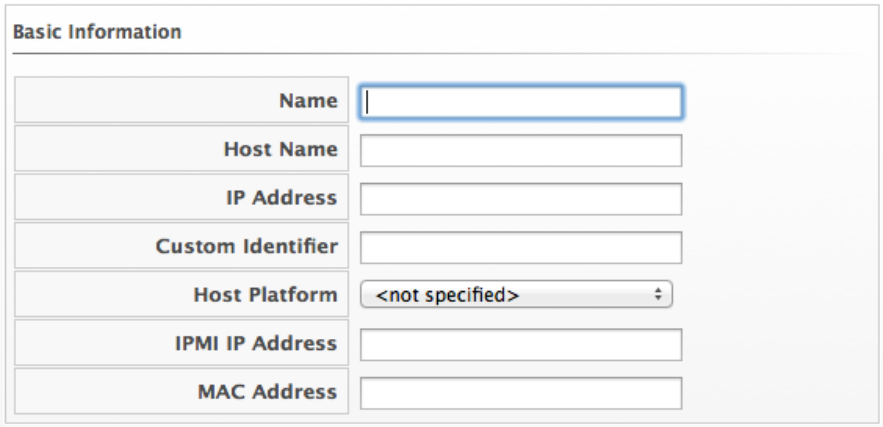
Adding a Single Host

To analyze a host for more detailed information, add it to the VMDB first. If the host has not been found during **Host Discovery** or **Provider Refresh**, and the host's IP address is known, use the **Add a New Host** button.

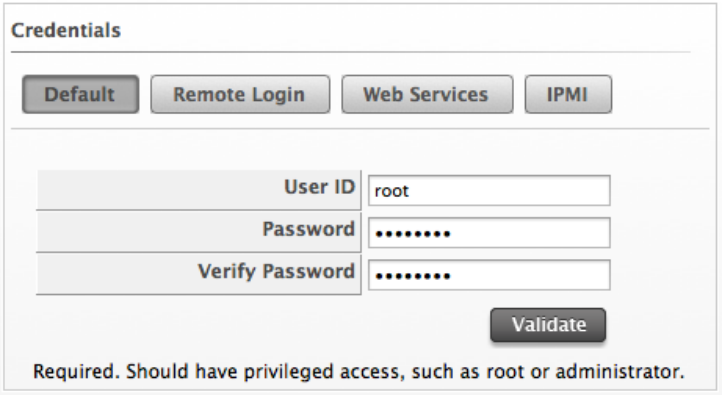
1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click **Configuration**, then click (../images/1862.png)**Add a New item**.
3. Type the **Name**, **Host Name**, and **IP Address** of the host to add. **Name** is how the device is labeled in the console. Select the type of operating system from the **Host Platform** dropdown. If the Host has been found during **Discovery** or **Refresh** and the host's operating

system has been identified, the **Host Platform** selector remains disabled. If adding an IPMI server for provisioning, add in the IP address of that host.

Important: The **Host Name** must use a unique fully qualified domain name.



4. In the **Credentials** box, the **Default** tab provides fields to type a user name with elevated security credentials and the user's password. If using domain credentials, the format for **User ID** is in the format of `[domainname]\[username]`. On ESX hosts, if the `SSH` login is disabled for the **Default** user, type in a user with remote login access on the **Remote Login** tab.

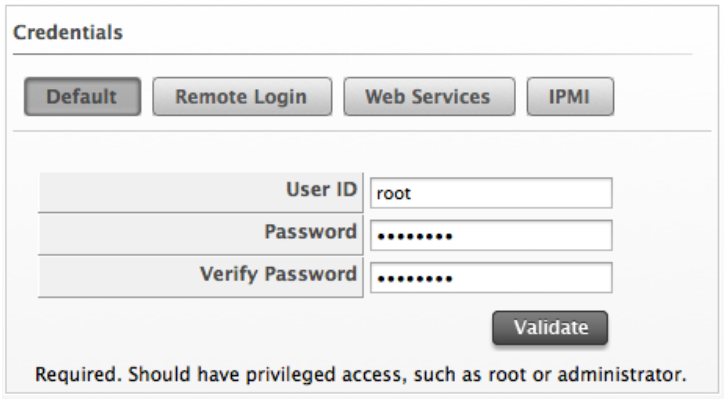


5. Click **Validate** to check the credentials.
6. Click **Save**.

Editing Hosts

If multiple hosts have the same settings or credentials, edit them at the same time.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click **Configuration**.
3. Check the Hosts to edit.
4. Click (**Edit Selected items**).
5. Use **Credentials** to provide login credentials required for this host.



- On the **Default** tab, type a user name with elevated security credentials and the users password. If you are using domain credentials, the format for User ID must be in the format of `[domainname]\[username]`.
- On `ESX` hosts, if `SSH` login is disabled for the **Default** user, type in a user with remote login access on the **Remote Login** tab. If this is not supplied, *Default* credentials will be used.
- Use **Web Services** to supply credentials for any web service calls made directly to the host system. If this is not supplied, **Default** credentials are used.

  Note:

  Login credentials are required for performing SmartState Analysis on the host's virtual machines and templates.

  For each type of credential used, the following information is required:

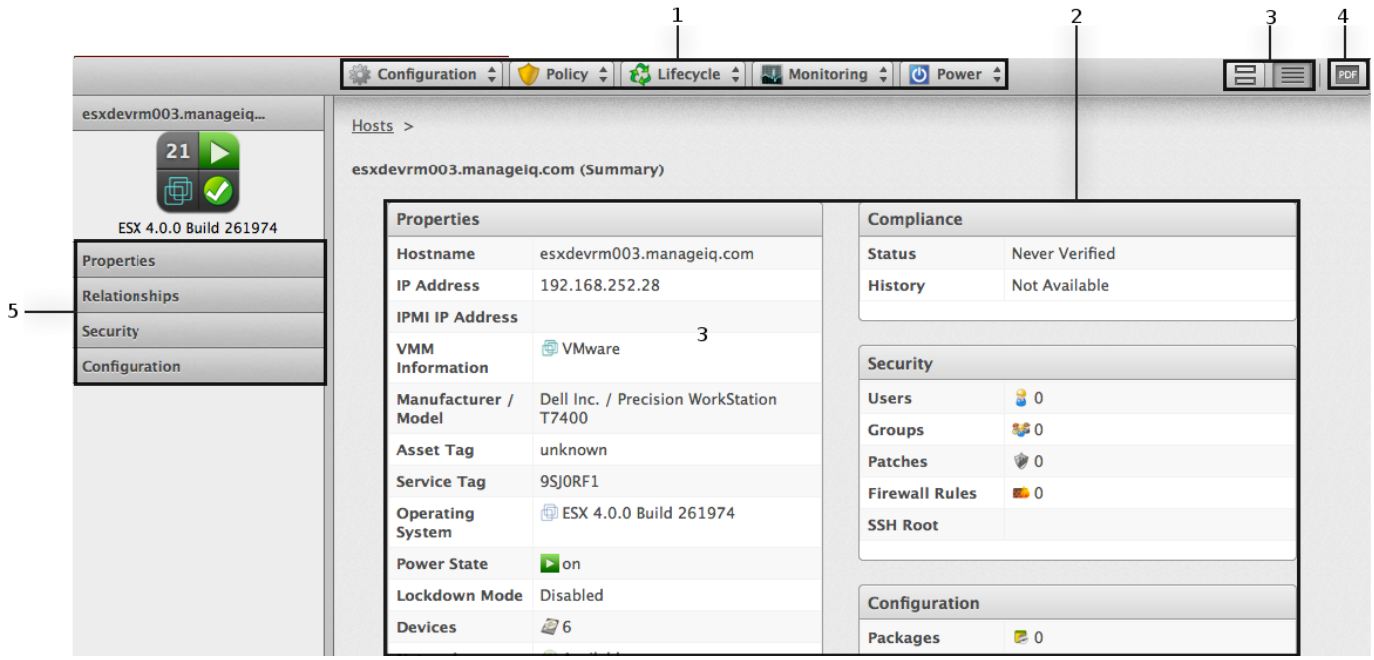
- Use **User ID** to specify a login ID.
- Use **Password** to specify the password for the User ID.
- Use **Verify Password** to confirm the password.

6. Test the credentials by using the *Select Host to validate against* drop down and click **Validate**.
7. Click **Save**.

Viewing a Host

You can click on a specific host to review it. The screen shows a host virtual thumbnail, a host taskbar, a host accordion, and a host summary.

Host Management Screen.

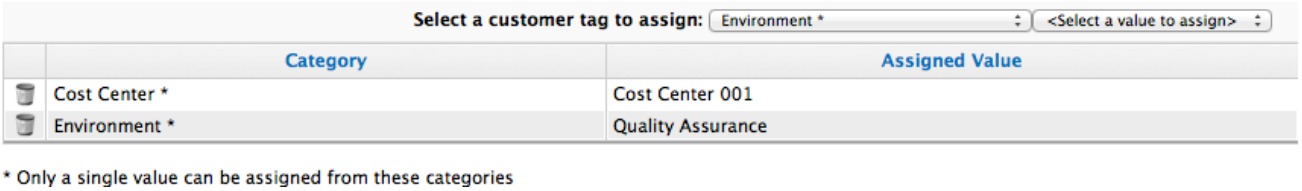


1. **Host Taskbar**: Use the host taskbar to take actions on the selected host
2. **Host Summary**: Use the host summary to see the properties of a host, drill down to a host's information, and view its installed virtual machines
3. **Host Summary Views**: Choose between graphical or text view of the provider summary
4. **Host PDF**: Generates host summary in PDF format
5. **Host Accordion**: See details about **Properties**, **Relationships**, **Security** and **Configuration** for the selected host

Tagging Multiple Hosts

To categorize hosts together, apply tags to multiple hosts at the same time.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Check the hosts to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. Select a customer tag from the first dropdown, and then a value for the tag.



5. Select more tags or click **Save** to save your changes.

Removing Hosts

If a host is decommissioned or requires troubleshooting, it might require removal from the VMDB.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Check the hosts to remove.

3. Click **Configuration**, and then (**Remove items from the VMDB**).
4. Click **OK**.

The hosts are removed. The virtual machines remain in the VMDB, but are no longer associated with their respective hosts.

Scaling Down Compute Hosts

Through Infrastructure Automation, you can perform a *Compute scale down* on a Red Hat OpenStack infrastructure provider. This process involves decreasing its Compute nodes used by an OpenStack infrastructure provider. Doing so involves putting a Compute node into *maintenance mode* and removing it from the provider afterwards. Once a node is in maintenance mode, it can be repurposed (for examle, re-provision it as a Controller node), repaired, or decommissioned altogether.

Before scaling down, evacuate or migrate any instances hosted on the node you are removing. For instructions on either procedure, see Migrating a Live Instance or Evacuating an Instance.

After migrating or evacuating instances from the node, set the node to maintenance mode. To do so:

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click the OpenStack compute node to be removed from the provider.
3. Click **Configuration**, and then (**Toggle Maintenance Mode**).

   Note:

   This option can only be used with OpenStack providers with at least two Compute nodes.

Repeat this procedure for every node you want to remove from the cloud provider.

After setting a Compute node to maintenance mode, you can scale down its provider:

1. Browse to menu: **Compute > Infrastructure > Providers**.
2. Click the provider to be scaled down.
3. Click **Configuration**, and then (**Scale Down**).
4. From the **Scale Infrastructure Provider Down** section, check the nodes to be removed from the provider. You can only do this for nodes where **Maintenance** is set to **true**.
5. Click **Scale Down**.

Refreshing Relationships and Power States for a Host

Refresh the relationships and power states of the items associated with your hosts from the Host Taskbar.

Note:

`root` or administrator credentials are required to get patch information.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click on the host to refresh.
3. Click **Configuration**, and then (**Refresh Relationships and Power States**) on the Host Taskbar.

Infrastructure Automation determines the state (running, stopped, or paused) of all virtual machines registered to the host.

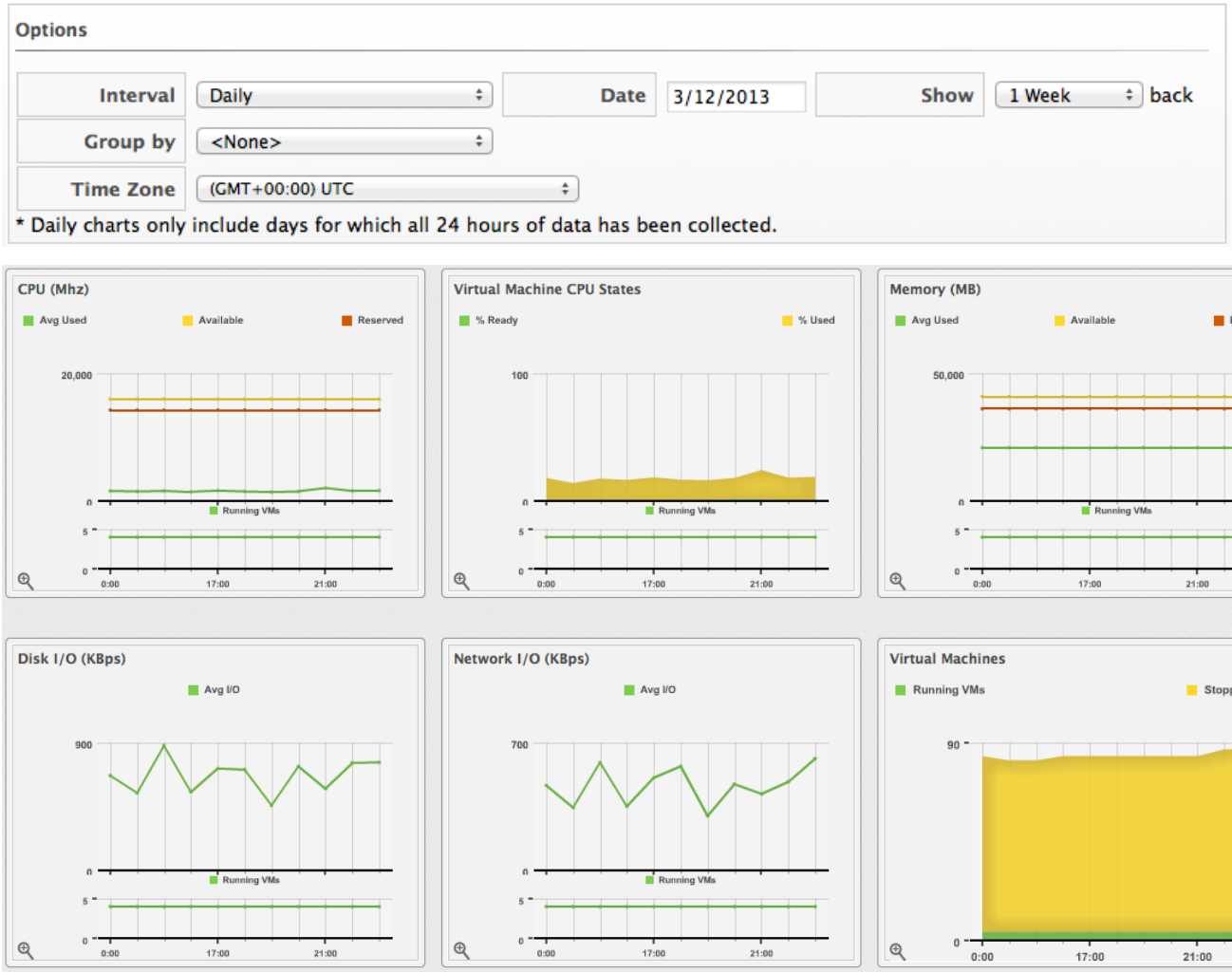
Viewing Capacity and Utilization Charts for a Host

View Capacity & Utilization data for hosts that are part of a cluster.

Note:

Your Infrastructure Automation server requires network visibility to the provider assigned the **Server Role** of **Capacity & Utilization Collector** to enable this feature.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click the Host to view capacity data.
3. Click (**Monitoring**), and then (**Utilization**) or from the Host accordion, click **Properties > Capacity & Utilization**.
4. From **Interval**, select to view hourly or daily data points and the dates to view data. Use **Group by** to group the lines by SmartTags. Use **Time Profiles** to select a time range for the data.



The charts are displayed for CPU, memory, disk, network, and running virtual machines.

Note:

Daily charts only include full days of data. If a day does not include all the 24 data points for a day, the data does not show for that day.

For information about data optimization including utilization trend reports, see Data Optimization.

Viewing the Host Timeline

View the timeline of events for the virtual machines registered to a host.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click the host to view the timeline.
3. Click (**Monitoring**), and then (**Timelines**) or from the host accordion, click **Properties > Timelines**.

4. From **Options**, customize the period of time to display and the types of events to see.
    - Use **Show** to select types of events to show on the timeline.
    - Use the **Interval** dropdown to select hourly or daily data points.
    - Use **Date** to type the date the timeline displays.
    - If you select to view a daily timeline, use **Show** to set how many days back to go. The maximum history is 31 days. If selecting **Hourly**, select the interval to see.
    - From the **Level** dropdown, select either a **Summary** event or a **Detail** list of events. For example, the detail level of a **Power On** event might include the power on request, the starting event, and the actual **Power On** event. If you select **Summary**, only the Power On event appears in the timeline.
    - The three **Event Group** dropdowns allow selection of different groups of events to display. Each group has its own color.
5. To see more detail on an item in the timeline, click on it. A balloon appears with a clickable link to the resource.

Host Virtual Summary

Clicking on a specific host shows the Host's Virtual Thumbnail and an **`operating system-sensitive`** screen of host information, called the Host Summary. Where applicable, click on a subcategory of the Host Summary to see more detail on that section.

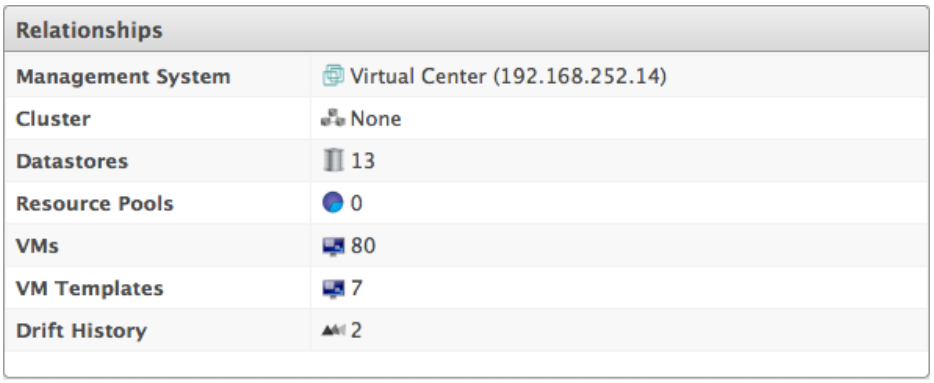
A **Refresh** provides some basic information on the Host. To get more detail, enter credentials for the host and perform a SmartState Analysis.

The Summary divides into the following categories.

- **Properties** include information such as base operating system, hostname, IP addresses, devices attached to the system, and storage adapters. Some categories can be clicked on for additional detail. For example, click **Network** to view the network adapters connected to the host.

Properties	
Hostname	per410a-t5.manageiq.com
IP Address	192.168.252.4
IPMI IP Address	
VMM Information	VMware
Manufacturer / Model	Dell Inc. / PowerEdge R410
Asset Tag	unknown
Service Tag	BBR8TK1
Operating System	ESXi 4.0.0 Build 504850
Power State	on
Lockdown Mode	Disabled
Devices	6
Network	Available
Storage Adapters	5
Number of CPUs	2
Number of CPU Cores	8
CPU Cores Per Socket	4
Memory	40 GB
Manangement Engine GUID	a55d2e44-8828-11e2-9abc-005056af00b4

- **Relationships** include information on the provider, cluster, datastores, resource pools, and installed virtual machines.



- **Security** shows the number of users, groups, patches installed, and firewall rules on the host. Click on any of these items to see further details.

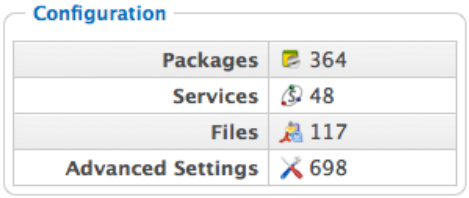
  Note:

  Run a SmartState Analysis on the host to retrieve this information.

- **Storage Relationships** shows the relationship the host has to LUNs, volumes, and file shares. The **Storage Inventory Role** must be enabled in the zone for these items to be populated.
- **Configuration** shows the number of packages and services installed. Click on any of these items to see more details.

  Note:

  Run a SmartState Analysis on the host to retrieve this information.



- **Smart Management** shows all tags assigned to this host.
- **Authentication Status** shows all the types of credentials entered for this host and the whether those credentials are valid.

Viewing Host Device Information

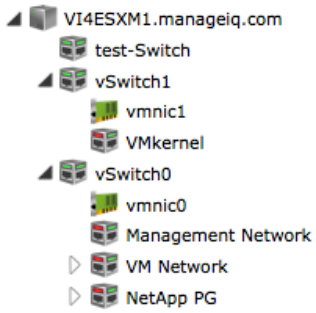
Access information on the hardware devices including processor, CPU type and speed, and memory for each host.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click the host to view the network information.
3. From the host accordion, click menu: **Properties > Devices**.

Viewing Host Network Information

Access information on networking including switches, network interfaces, and local area networks for each host.

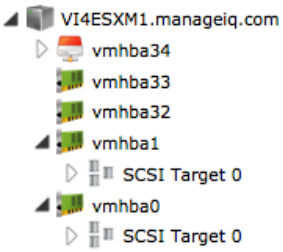
1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click the host to view the network information.
3. From the host accordion, click menu: **Properties > Network**.



Viewing Storage Adapters

Access information on the storage adapters including storage type for each host.

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click the host to view the network information.
3. From the host accordion, click menu: **Properties > Storage Adapters**.



Detecting Drift on Hosts

Over time, the configuration of a host might change. Drift is the comparison of a host to itself at different points in time. The host requires analysis at least twice to collect information. Detecting drift provides you the following benefits:

- See the difference between the last known state of a host and its current state.
- Review the configuration changes that happen to a particular host between multiple points in time.
- Capture the configuration drifts for a single host across a time period.

Detect drift on hosts:

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Click on the host to view drift.
3. Click **Relationships** in the host accordion.
4. Click **Drift History**.
5. Check the analyses to compare.
6. Click (**Drift**) at the top of the screen. The results display.
7. Check the **Comparison** sections on the left to view in your comparison.
8. Click **Apply**.
9. Click the plus sign next to the sections name to expand it.
    - An item displayed on red text shows a change from the base analysis. An item displayed in black text shows no change from the base analysis.

- A (**Changed from previous**) shows a change since the last analysis.
- A (**Same as previous**) means no change since the last analysis.
- Click (**Remove from drift**) at the bottom of a column to remove a specific analysis. The drift recalculates and the new results display.

10. Click (**Expanded View**) to see the expanded view. Click (**Compressed View**) to compress the information.
11. Click the minus sign next to the sections name to collapse it.
12. To limit the type of views, you have three buttons in the taskbar:
    - Click (**All attributes**) to see all attributes of the sections you selected.
    - Click (**Attributes with different values**) to see only the attributes that are different across the drifts.
    - Click (**Attributes with the same values**) to see only the attributes that are the same across drifts.

The drift comparison displays. Download the data or create a report from your drift for analysis using external tools.

Creating a Drift Report for Hosts

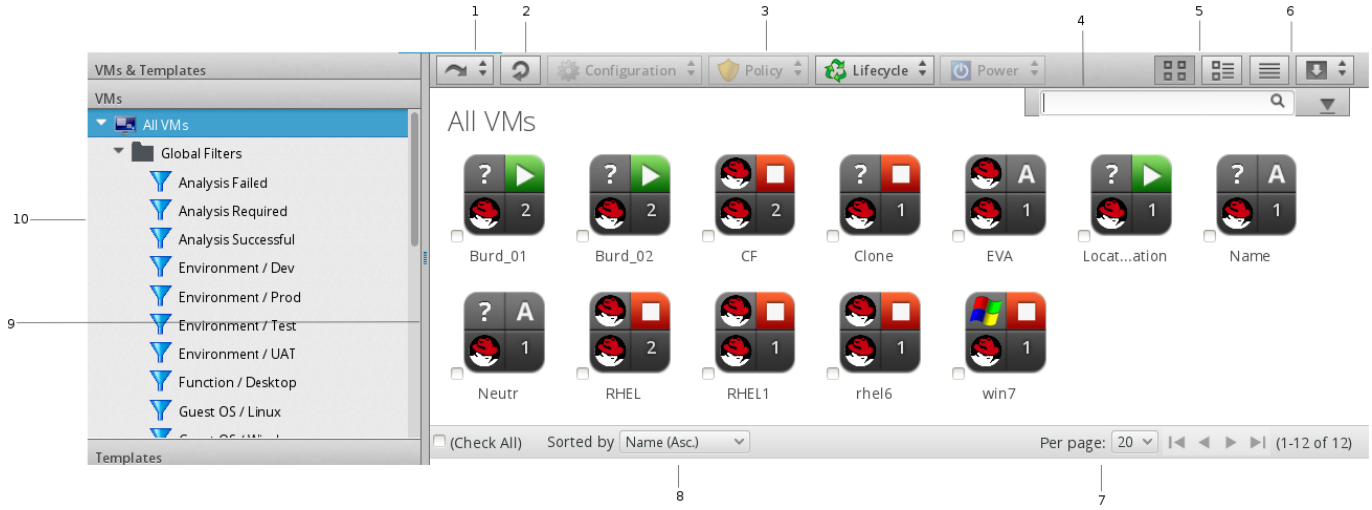
Use the drift report feature to export information about your host's drift.

1. Create the comparison to analyze.
2. Click (**Download**).
3. Click the output button for the type of report you want.
    - Click (**Download drift report in TXT format**) for a text file.
    - Click (**Download drift report in CSV format**) for a comma-separated file.
    - Click (**Download drift report in PDF format**) for a PDF file.

Virtual Machines

The heterogeneous virtual machine container and guest support combined with the ability to analyze information inside the virtual machine - such as disk space, patch level or installed applications - provides in-depth information across the virtual environment. This rich set of information enables Infrastructure Automation users to improve problem resolution times and effectively manage virtual machines.

The **Virtual Machines** pages display all virtual machines that were discovered by your server. Note that if you have applied a filter to a user, it will be in effect here. The **Virtual Machines** taskbar is a menu driven set of buttons that provide access to functions related to virtual machines.



1. History button
2. Refresh screen button
3. Taskbar
4. Name search bar/Advanced Search button
5. View buttons
6. Download buttons
7. Navigation bar
8. Sort dropdown
9. Main area in Grid View
10. Provider/Filter Navigation

The console uses **Virtual Thumbnails** to describe virtual machines and templates. Each thumbnail contains four quadrants by default. This allows you to glance at a virtual machine for a quick view of its contents.

1. Top left quadrant: Operating system of the Virtual Machine
2. Bottom left quadrant: Virtual Machine Hosts software
3. Top right quadrant: Power state of Virtual Machine or Status icon
4. Bottom right quadrant: Number of Snapshots for this Virtual Machine

Icon	Description
T	Template: Virtual Template
R	Retired: When a virtual machine or instance is no longer required, it can be retired. Once a virtual machine or instance reaches its retirement date and time, it is immediately shut down and not allowed to restart. If an attempt to restart is made, Infrastructure Automation will shut down the virtual machine or instance.
A	Archived: An archived virtual machine has no host or datastore associated with it. Archiving is done to move virtual machines to a low cost storage, either on demand or during retirement, if requested, to avoid incurring extra cost on a virtualized infrastructure due to virtual machine sprawl.
O	Orphaned: An orphaned virtual machine has no host but has a datastore associated with it. Orphaned virtual machines are those that have been removed from their providers but still exist on the storage. An orphaned virtual machine is unable to identify the associated host. A virtual machine also shows as orphaned if it exists on a different host than the host expected by the provider's server.
D	Disconnected: A disconnected virtual machine is one that has lost connection to either the provider's storage, host, or both. A disconnect is usually a result of network issues on the provider side. For instance, if during virtual machine provisioning the storage is not set up or deleted, the virtual machine will still exist on the provider, but will not run on the host as it has lost connection to its provider's storage.

Icon	Description
	On: Virtual Machine is powered on.
	Off: Virtual Machine is powered off.
	Suspended: Virtual Machine has been suspended.

The **Virtual Machines** page has three accordions organizing your virtual machines and templates in different ways. All of these accordions share a set of common controls:

- Use **VMs and Templates** to view your virtual machines and templates organized by Provider. In addition, you can see archived and orphaned items here.
- Use **VMs** to view, apply filters, and collect information about all of your virtual machines.
- Use **Templates** to view, apply filters, and collect information about all of your templates.

Through the console, you are able to view your virtual machines in multiple ways. For your virtual machines, you can:

- Filter virtual machines
- Change views
- Sort
- Create a report
- Search by MyTags
- Search by collected data

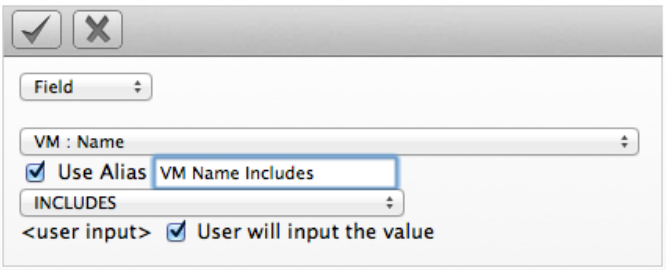
Filtering Virtual Machines and Templates

The **Virtual Machine Filter** accordion is provided so that you can easily navigate through groups of virtual machines. You can use the ones provided or create your own through **Advanced Filtering** capabilities.

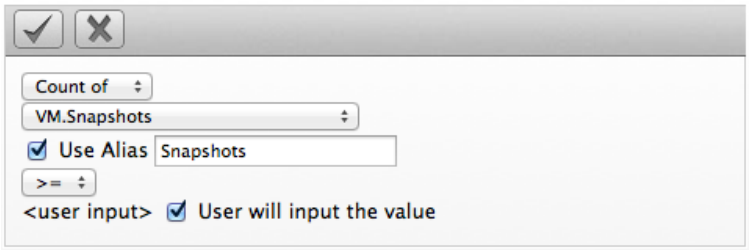
1. Navigate to **Compute > Infrastructure > Virtual Machines**.
2. Go to the **VMs** or **Templates** accordion.
3. Click on the desired filter from the left pane.

Creating a Virtual Machine or Template Filter

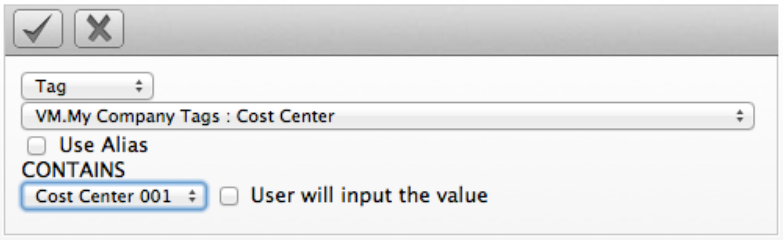
1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Go to the **VMs** or **Templates** accordion.
3. Click **All VMs** or **All Templates**, then click (**Advanced Search**) to open the expression editor.
4. Use the expression editor to choose the appropriate options for your criteria. Based on what you choose, different options will show.
   - For all of the types of searches, you have the options of creating an alias and requested user input. Select **Use Alias** to create a user friendly name for the search. If you are requested user input for the search, this text will show in the dialog box where the input is requested.
   - Click **Field** to create criteria based on field values.



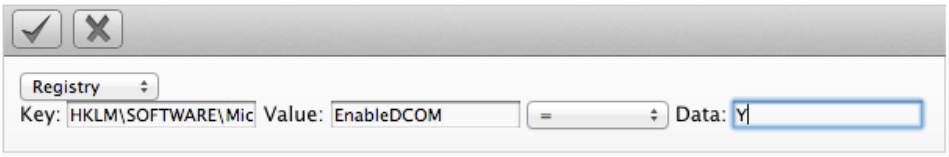
   - Click **Count of** to create criteria based on the count of something, such as the number of snapshots for a virtual machine, or the number of virtual machines on a host.



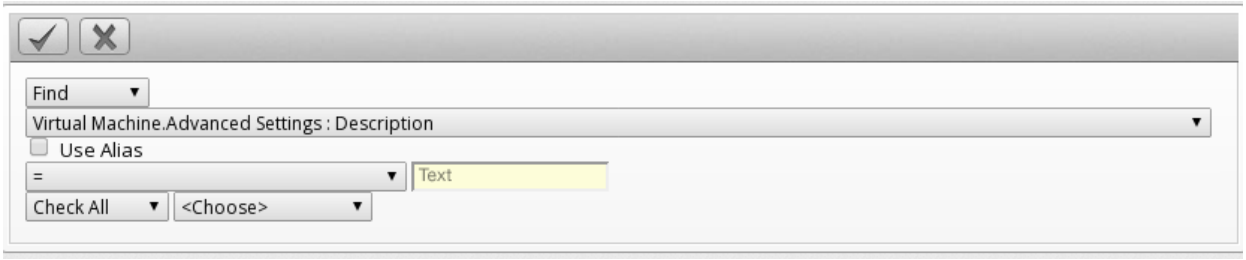
- Click **Tag** to create criteria based on tags assigned to your virtual infrastructure, such as for power states or production tagging.



- Click **Registry** to create criteria based on registry values, such as the DCOM status of a Windows system. Note this criteria applies only to Windows operating systems.



- Click **Find** to seek a particular value, and then check a property.



5. Click ✓ (**Commit Expression Element Changes**) to add the expression.
6. Click **Save**.
7. Type in a name for the search expression in **Save this VM** search as. (Note that this title depends on the type of resource you are searching.) To set the filter to show globally, check **Global search**.
8. Click **Save**.

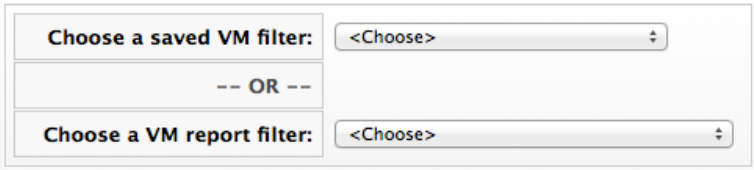
The filter is saved and will show in the **My Filters** area of the **Filter** accordion. If you checked **Global search**, the filter will show under **Global Filters**.

Loading a Report Filter or Search Expression

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to search either **VMs** or **Templates**.
3. Click » (**Advanced Search**) to open the expression editor.
4. Click **Load**.
5. Select either a saved virtual machine search or a virtual machine report filter.

   Note:

The set of items to select will depend on the type of resource you are searching.



6. Click **Load** to load the search expression.
7. If you want to edit the expression, click on it and make any edits for the current expression.
   - Click (**Commit expression element changes**) to add the changes.
   - Click (**Undo the previous change**) to remove the change you just made.
   - Click (**Redo the previous change**) to put the change that you just made back.
   - Click AND (**AND with a new expression element**) to create a logical AND with a new expression element.
   - Click OR (**OR with a new expression element**) to create a logical OR with a new expression element.
   - Click NOT (**Wrap this expression element with a NOT**) to create a logical NOT on an expression element or to exclude all the items that match the expression.
   - Click (**Remove this expression element**) to take out the current expression element.
8. Click **Load**.
9. Click **Apply**.

Changing Views for Virtual Machines and Templates

While you can set the default view for different pages from the settings menu, then **Configuration > My Settings > Default Views**, the current view can also be controlled from the Virtual Machines pages.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to view.
3. Click the appropriate button for the desired view.
   - Click for **Grid View**.
   - Click for **Tile View**.
   - Click for **List View**.

Sorting Virtual Machines and Templates

Virtual machines and templates can be sorted by Name, Cluster, Host, Datastore, Compliance, Last Analysis Time, Total Snapshots, or Region.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to sort.
3. To sort virtual machines or templates when in grid or tile view:
   - From the **Sort by** dropdown, click the attribute to sort.
4. To sort virtual machines or templates when in list view:
   - Select the **List View**.

- Click on the **Column Name** to sort. For example, click on **Cluster** to sort by the name of the cluster.

Creating a Virtual Machine or Template Report

For a listing of virtual machines and templates, you can create a quick report in CSV, TXT, or PDF formats.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items for report creation.
3. Click (**Download**).
   - Click for a **TXT** file.
   - Click for a **CSV** file.
   - Click for a **PDF** file.

Searching for Virtual Machines or Templates

To the right of the taskbar on the *Virtual Machines* page, you can enter names or parts of names for searching. You can search in the following ways:

- Type characters that are included in the name. For example, if you type **sp1**, all Virtual Machines whose names include **sp1** appear, such as **Windows2003sp1** and **Sp1clone**.
- Use * at the end of a term to search for names that begin with specific characters. For example, type **v*** to find all virtual machines whose names begin with the letter **v**.
- Use ***** at the beginning of a term to search for names that end with specific characters. For example, type ***sp2** to find all virtual machines whose names end with **sp2**.
- Erase all characters from the search box to go back to viewing all virtual machines.

Search for virtual machines or templates:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to search.

3. In the **Name Filter** bar in the upper right corner of the window, type your criteria.
4. Click (**Search by Name within results**) or press **Enter**.
5. Type in other criteria to filter on what is currently displayed.
6. Click (**Search by Name within results**) or press **Enter**.

Analyzing Virtual Machines and Templates

Analyze a virtual machine to collect metadata such as user accounts, applications, software patches, and other internal information. If Infrastructure Automation is not set up for automatic analysis, perform a manual analysis of a virtual machine. To perform a SmartState analysis, Infrastructure Automation requires a running SmartProxy with visibility to the virtual machine's storage location. If the virtual machine is associated with a host or provider, ensure the virtual machine is registered with that system to be properly analyzed; the server requires this information since a snapshot might be created.

Note:

SmartState Analysis of a virtual machine requires access to its host. To perform a successful analysis, edit the virtual machine's host and enter the host's authentication credentials.

1. Navigate to **Compute > Infrastructure > Virtual Machines**.

2. Click the accordion for the items to analyze.
3. Check the **Virtual Machines** and **Templates** to analyze.
4. Click (**Configuration**), then (**Perform SmartState Analysis**).
5. Click **OK**.

Red Hat Enterprise Virtualization Prerequisites

SmartState Analysis on Red Hat Enterprise Virtualization Manager 3.1 and Above - Storage Support Notes

Note the following requirements when performing SmartState Analysis on Red Hat Enterprise Virtualization Manager 3.1 and above.

- NFS
  - The Infrastructure Automation appliance requires a mount to the NFS datastore.
- iSCSI / FCP
  - For Red Hat Enterprise Virtualization 3.1 and 3.2, clusters must use full Red Hat Enterprise Linux hosts and not Red Hat Enterprise Virtualization Hypervisor hosts. You can use either type of host in Red Hat Enterprise Virtualization 3.3 and above.
  - Each Infrastructure Automation appliance performing SmartState Analysis requires sharable, non-bootable DirectLUN access to each attached iSCSI/FCP storage domain. In order to perform smart analysis, the appliance must mount the data storage as a DirectLUN disk.
  - A Infrastructure Automation appliance **must** reside in each datacenter with the iSCSI / FCP storage type.
- Other Notes
  - The **Edit Management Engine Relationship** option enables the VM SmartState Analysis job to determine the datacenter where a Infrastructure Automation appliance is running and thus to identify what storage it has access to in a Red Hat Enterprise Virtualization environment.
    - After setting up a Infrastructure Automation appliance and performing a refresh of the provider, find the Infrastructure Automation appliance in the **Virtual Machine** accordion list and view its summary screen.
    - Click **Configuration > Edit Management Engine Relationship**.
    - Select the server that relates to this instance of the Infrastructure Automation appliance.

If you attach a DirectLUN disk after configuring the Infrastructure Automation database, access the appliance in a terminal and run `pvscan` to detect the DirectLUN disk. Alternatively, in Infrastructure Automation 5.2.1 and above, you can restart the appliance to detect the disk automatically.

SmartState Analysis on Red Hat Enterprise Virtualization Manager 3.0 - Storage Support Notes

There are two additional steps required to perform a SmartState Analysis on Red Hat Enterprise Virtualization Manager 3.0 using **`iSCSI`** or **`FCP`** storage. NFS storage does not have these requirements.

1. Enable **`DirectLUN`** support for the host and Infrastructure Automation appliance that performs the analysis.
   - Enable **`DirectLUN`** on host.
   - Enable **`DirectLUN`** on the Infrastructure Automation appliance. To do this, edit the desired Red Hat Enterprise Virtualization storage and get the **`LUNID`** value. Then, on the Infrastructure Automation appliance virtual machine in the Red Hat Enterprise Virtualization user interface, right-click and select **Edit > Custom Properties** and enter the following in the **Custom Properties** edit box:

     `directlun=<LUN ID>:readonly`

     If you have multiple storage domains, separate them by a comma, similar to:

     `directlun=<LUN ID 1>:readonly,<LUN ID 2>:readonly,<LUN ID N>:readonly`

Note:

The Infrastructure Automation appliance must reside in the same data center as the storage you are trying to connect. If you have multiple data centers with `iSCSI` or `FCP` storage, you need a Infrastructure Automation appliance in each data center to support virtual machine scanning.

2. **Set Server Relationship** - This is required to allow the virtual machine SmartState analysis job to determine which data center a Infrastructure Automation appliance is running and therefore identify what storage it has access to in a Red Hat Enterprise Virtualization environment.

    1. After setting up a Infrastructure Automation appliance and performing a refresh of the Provider, find the Infrastructure Automation appliance in the **Virtual Machine** accordion list and view its summary screen.
    2. Click (**Configuration**), and then (**Edit Server Relationship**).
    3. Select the server that relates to this instance of the Infrastructure Automation appliance.

VMware vSphere Prerequisites

Comparing Virtual Machines and Templates

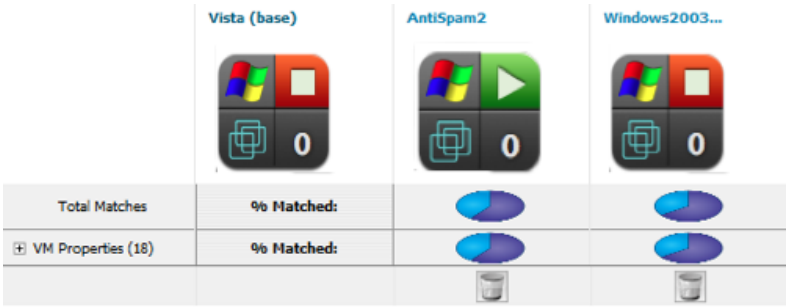
The Infrastructure Automation Server allows you to compare multiple virtual machines. This allows you to see how different virtual machines are from their original template. This helps detect missing patches, unmanaged user accounts, or unauthorized services.

Use the comparison feature to:

- Compare multiple virtual machines from different hosts.
- Compare multiple virtual machines side-by-side.
- Quickly see similarities and differences among multiple virtual machines and a base.
- Narrow the comparison display to categories of properties.
- Print or export in the comparison results to a PDF or CSV file.

Compare virtual machines and templates:

1. Navigate to **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to analyze.
3. Check the items to compare.
4. Click btn:[(Configuration)], and then btn:[(Compare Selected items)]. The comparison displays in a compressed view with a limited set of properties listed.



5. To delete an item from the comparison, click btn:[(Remove this item from Inventory)] at the bottom of the items column. This option is only available when comparing more than two virtual machines.
6. To view many items on one screen, go to a compressed view by clicking btn:[(Compressed View)]. To return to an expanded view, click btn:[(Expanded View)].
7. To limit the mode of the view, there are two buttons in the task bar.
    - Click btn:[(Details Mode)] to see all details for an attribute.

- Click btn:[(Exists Mode)] to limit the view to if an attribute exists compared to the base or not. This only applies to attributes that can have a boolean property. For example, a user account exists or does not exist, or a piece of hardware that does or does not exist.

8. To change the base virtual machine that all the others are compared to, click its label at the top of its column.
9. To go to the summary screen for a virtual machine, click its btn:[Virtual Thumbnail] or icon.

Virtual Machine and Templates Comparison Sections

The following table describes the different sections for comparison information.

Section	Description
Properties	Use this section to see basic information on the file location of the virtual machine, its name, and the virtual machine monitor vendor. Hardware, disk, CD/DVD drives, floppy drive, network adapter, and volume information is also included.
Security	Use this to see users and groups for the virtual machine, including those which may be unauthorized compared to a template.
Configuration	Use this to see Guest Applications, Win32 services, Linux Init Processes, Kernel Drivers, File System Drivers, and Patches.
My Company Tags	Use this to see all tags.

Using the Virtual Machine Comparison Sections

Use the comparison sections to view various comparison data and display the data in different ways.

1. On the left of a comparison screen, select what categories of properties to display.
2. Click **Apply**.
3. Click the plus sign next to the sections name to expand it.
4. The following descriptions pertain to the **Expanded View** . Whether you see the value of a property or an icon representing the property depends on the properties type.
    - A property displayed in the same color as the base means that the compared virtual machine matches the base for that property.
    - A property displayed in a different color from the base means that the compared virtual machine does not match the base for that property.
5. If you are in the **Compressed View** , the values of the properties will not be displayed. The icons shown below will describe all items.
    - A (**checkmark**) means that the compared virtual machine matches the base for that property. If you hover over it, the value of the property will display.
    - A (**x**) means that the compared virtual machine does not match the base for that property. If you hover over it, the value of the property will display.
6. Click the minus sign next to the sections name to collapse it.

Your comparison can be viewed in multiple ways. Export the data or create a report from your comparison for analysis using external tools.

Creating a Virtual Machine Comparison Report

Output the data from a comparison report in TXT, CSV or PDF formats.

1. Create the comparison for the report.
2. Click the output button for the chosen report type.
    - Click (**Download comparison report in TXT format**) for a text file.
    - Click (**Download comparison report in CSV format**) for a CSV file.

- Click (**Download comparison report in PDF format**) for a PDF file.

Controlling the Power State of Red Hat Virtualization Virtual Machines

Follow this procedure to control the power states of Red Hat Virtualization VMs through the Infrastructure Automation console.

1. Navigate to **Compute > Infrastructure > Virtual Machines**.
2. Click on a VM to change the power state.
3. Click **Power Operations**, then click the button for the desired power operation. Available operations will depend on the current power state of the VM.
   - Click (**Shutdown Guest**) to shutdown the guest OS on the VM.
   - Click (**Restart Guest**) to restart the guest OS on the VM.
   - Click (**Power On**) to power on the VM.
   - Click (**Power Off**) to power off the VM.
   - Click (**Suspend**) to suspend the VM.
   - Click (**Reset**) to reset the VM.
4. Click **OK**.

Refreshing Virtual Machines and Templates

Refresh your virtual machines to get the latest data the provider or host can access. This includes information such as the power state, container, and hardware devices attached to the virtual machine.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to analyze.
3. Check the items to refresh.
4. Click **Configuration**, and then (**Refresh Relationships and Power States**) on the **Virtual Machine Taskbar**.

The console returns a refreshed list of the data associated with the selected virtual machines.

Extracting Running Processes from Virtual Machines and Templates

Infrastructure Automation can collect processes running on Windows virtual machines. To do this, enter domain credentials for the zone where the virtual machine is located.

The virtual machine must be running and must have an IP address in the VMDB, usually obtained from a SmartState Analysis.

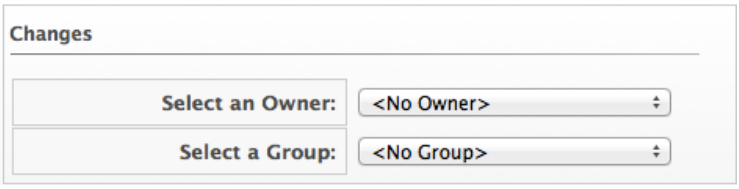
1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Check the Virtual Machines to collect the processes.
3. Click **Configuration**, and then (**Extract Running Processes**) on the Taskbar.
4. Click **OK**.

The server returns the running processes. View the summary of the virtual machine to see the details.

Setting Ownership for Virtual Machines and Templates

You can set the owner of a group of virtual machines and templates by either individual user or group. This allows you an additional way to filter and can be used to enforce quotas.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to change.
3. Check the items to set ownership.
4. Click **Configuration**, and then (**Set Ownership**) on the **Virtual Machine Taskbar**.
5. From the **Select an Owner** dropdown, select a user, and from the **Select a Group** dropdown, select a group.



6. Click **Save**.

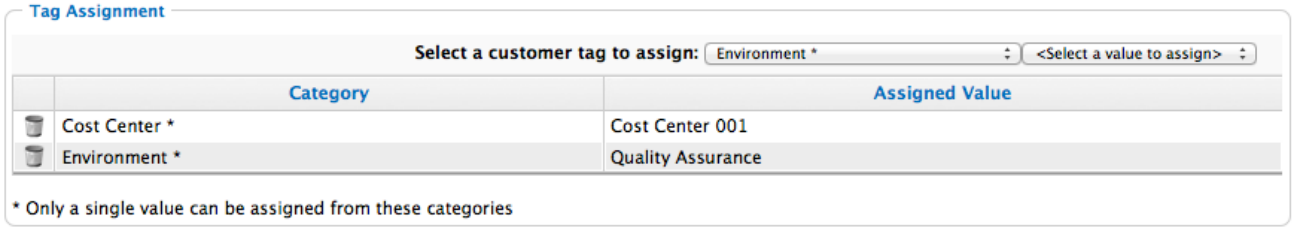
Removing Virtual Machines and Templates from the VMDB

If a virtual machine has been decommissioned or you need to perform some troubleshooting, you might need to remove a specific virtual machine from the VMDB. This does not however remove the virtual machine or template from its Datastore or Provider.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to remove.
3. Check the items to remove.
4. Click **Configuration**, and then (**Remove from the VMDB**) button.
5. Click **OK**.

Tagging Virtual Machines and Templates

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to tag.
3. Check the items to tag.
4. Click (**Policy**), and then (**Edit Tags**).
5. Select a customer tag from the first dropdown, and then a value for the tag.



Viewing a VMware Virtual Machine's Storage Profile

VMware storage profiles allow you to assign policies to datastores. Storage profiles are used to tag virtual machines to ensure they operate in compliance with settings in the datastore.

Infrastructure Automation retrieves VMware virtual machine storage profile information in the inventory, and associates the virtual machines and disks with them.

To view a virtual machine's storage profile:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click a VMware virtual machine to open its summary page.
3. The VMware **Storage Profile** is listed under **Properties**.

You can assign a storage profile when provisioning a VMware virtual machine in Infrastructure Automation, by using the virtual machine as a template to clone. See Provisioning Virtual Machines in *Provisioning Virtual Machines and Hosts* for instructions.

Viewing Running Processes after Collection

1. Click a virtual machine with collected processes.
2. From the **Diagnostics** area, click **Running Processes**.

	Name	PID	Memory %	Memory Usage	CPU %	CPU Time
	AppleMobileDeviceService.e	1,360	0.54	2,883,584	0.00	0
	EarthAgent.exe	1,412	1.36	7,278,592	0.00	409
	NSClient++.exe	1,524	1.70	9,093,120	0.00	176
	SoftwareUpdate.exe	2,028	3.37	18,075,648	0.00	0
	SpntSvc.exe	1,640	1.13	6,041,600	0.00	695
	StOPP.exe	2,052	0.50	2,658,304	0.00	0
	StWatchDog.exe	3,672	0.40	2,125,824	0.00	0
	VMwareService.exe	1,664	14.34	76,918,784	0.00	108
	VMwareTray.exe	2,368	0.14	749,568	0.00	0
	VMwareUser.exe	3,276	0.13	696,320	0.00	2
	csrss.exe	332	1.21	6,475,776	0.00	5
	csrss.exe	3,304	1.24	6,647,808	0.00	3

The most recent collection of running processes is displayed. Sort this list by clicking on the column headers.

Editing Virtual Machine or Template Properties

Edit the properties of a virtual machine or template to set parent and child virtual machines. SmartState Analysis also can detect this.

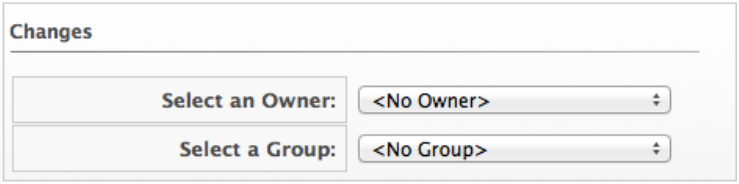
1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to edit.
3. Click the item to edit properties.
4. Click **Configuration**, and then (**Edit selected item**) on the Taskbar.
5. From the **Parent VM** dropdown, select the parent virtual machine.
6. From **Child VM** selection, select virtual machines that are based on the current virtual machine from the list of **Available VMs**.
7. Click **Save**.

Setting Ownership of a Virtual Machine or Template

Set the owner of a virtual machine or template by either individual user or group. This allows you an additional way to filter configuration items.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the items to analyze.

3. Click the item to set ownership.
4. Click **Configuration**, and then (**Set Ownership**) on the taskbar.
5. From the **Select an Owner** dropdown, select a user.



6. From the **Select a Group** dropdown, select a group.
7. Click **Save**.

Right Sizing a Virtual Machine

Infrastructure Automation uses collected statistics to recommend the best size for a virtual machine. Infrastructure Automation uses the information from the **Normal Operating Range** to calculate the recommendations.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click a virtual machine for right-sizing.
3. Click **Configuration**, and then (**Right-Size Recommendations**) button.

A new page appears with three levels of Memory and CPU recommendations, Conservative, Moderate, and Aggressive, next to the Normal Operating Range statistics.

Viewing Capacity and Utilization Charts for a Virtual Machine

You can view capacity and utilization data for virtual machines that are part of a cluster. Note that daily charts only include full days of data. If all 24 data points for a day are not available, daily charts are not displayed. For some capacity and utilization data, Infrastructure Automation calculates and shows trend lines in the charts which are created using linear regression. The calculation uses the capacity and utilization data collected by Infrastructure Automation during the interval you specify.

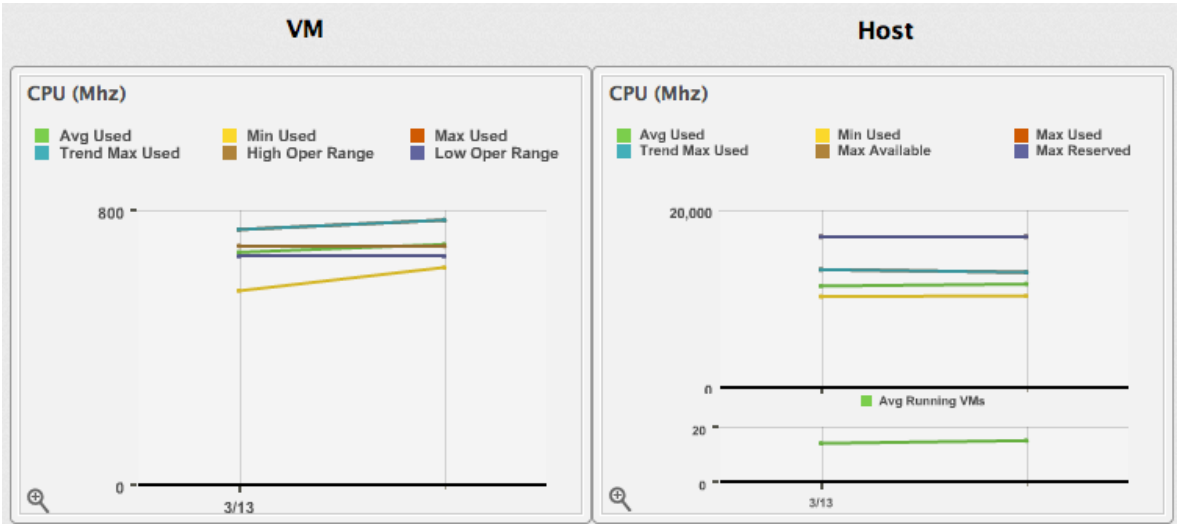
Note:

You must have a server with network visibility to your provider assigned the server role of **Capacity & Utilization Collector** to use this feature.

The virtual machine must be powered on to collect the data.

1. From **Compute > Infrastructure > Virtual Machines**, click the accordion that you want to view capacity data for.
2. Click the item you want to view.
3. Click (**Monitoring**), and then (**Utilization**).
4. From **Interval**, select to view **Daily**, **Hourly**, or **Most Recent Hour** data points. When choosing **Daily**, you can also select the **Date**, and how far back you want to go from that date. When selecting **Hourly**, you can select the date for which you want to view hourly data. If you are using **Time Profiles**, you will be able to select that as an option, also.

5. From **Compare to**, select **Parent Host** or **Parent Cluster**. The capacity and utilization charts for both items will show simultaneously.



Note:

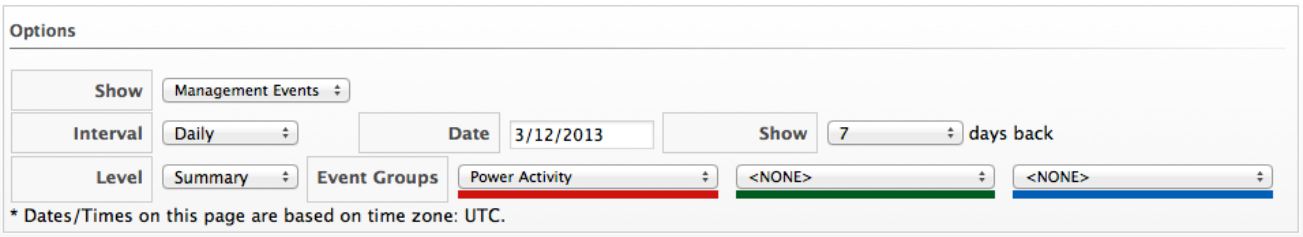
Daily charts only include full days of data. This means Infrastructure Automation does not show daily data for a day without a complete 24 data point range for a day.

For information about data optimization including utilization trend reports, see Data Optimization.

Viewing the Virtual Machine or Template Timeline

View the timeline of events for a virtual machine or template if registered to a Host.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the virtual machine to view the timeline.
3. Click (**Monitoring**), and then (**Timelines**) on the taskbar.
4. From **Options**, customize the period of time to display, and the types of events to view.



   - Use the **Interval** dropdown to select hourly or daily data points.
   - Use **Date** to type the date of the timeline to display.
   - If viewing a daily timeline, use **Show** to set how many days back to go. The maximum history is 31 days.
   - The three **Event Group** dropdowns allow selection of different event groups to display. Each has its own color.
   - From the **Level** dropdown, select either a **Summary** event or a **Detail** list of events. For example, the detail level of a **Power On** event might include the power on request, the starting event, and the actual **Power On** event. If you select **Summary**, you only see the **Power On** event in the timeline.
5. To see more detail on an item in the timeline, click on it. A balloon appears with a clickable link to the resource.

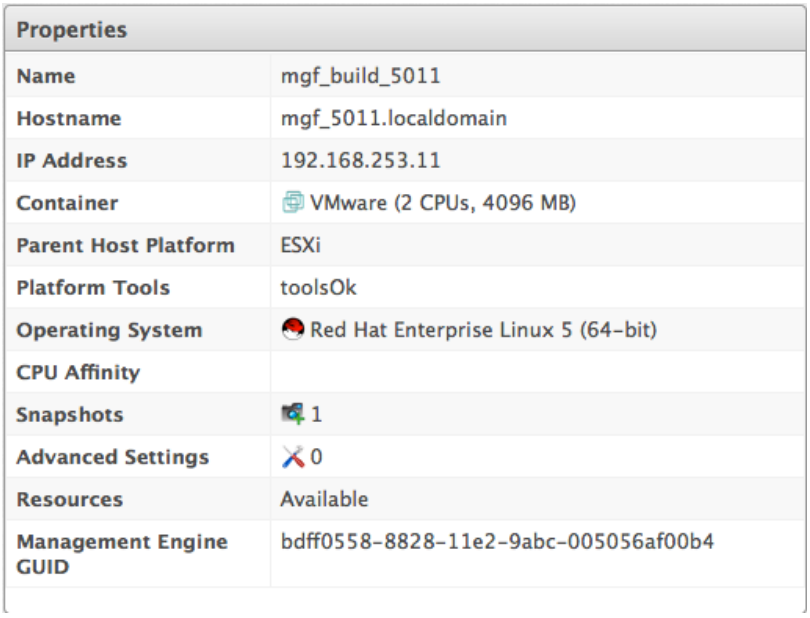
Virtual Machine or Template Summary

When you click on a specific virtual machine or template, you will see the **Virtual Thumbnail**, and an operating system-specific screen of the item, called the **Summary**. Where applicable, click on a subcategory of the **Summary** to see more detail on that section.

Note:

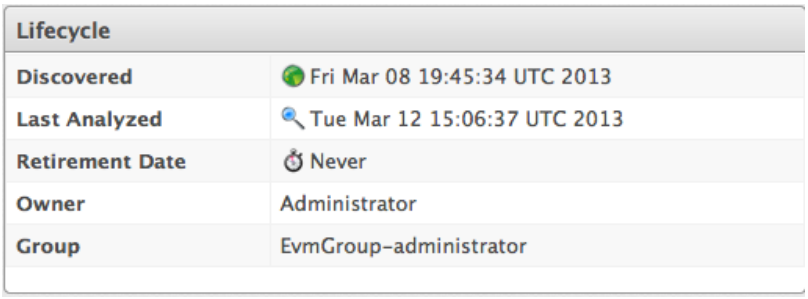
When you perform a SmartState Analysis on a virtual machine or template, you get more detailed information in these categories.

- **Properties** include information such as the base operating system, hostname, IP addresses, Virtual Machine vendor, CPU Affinity, devices attached to the system, and snapshots. This includes the ability to analyze multiple partitions, multiple disks, Linux logical volumes, extended partitions, and Windows drives. Some categories can be clicked on for additional detail. For example, click **Container** to view notes associated with a virtual machine.



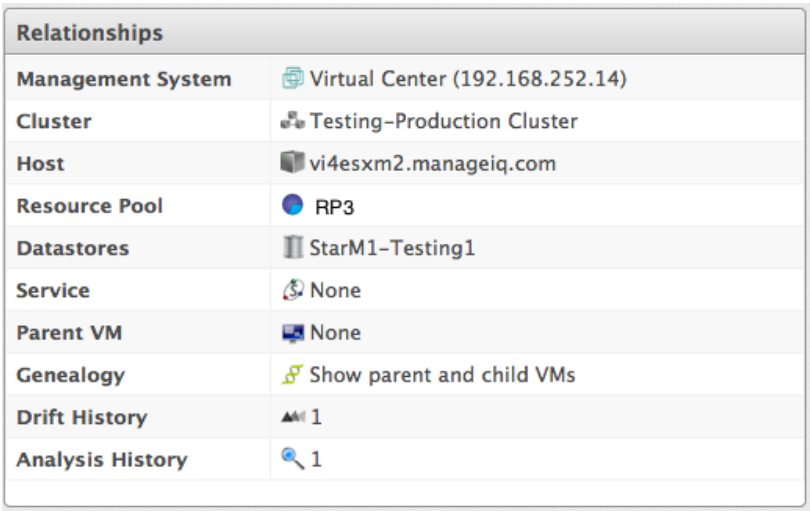
Properties	
Name	mgf_build_5011
Hostname	mgf_5011.localdomain
IP Address	192.168.253.11
Container	VMware (2 CPUs, 4096 MB)
Parent Host Platform	ESXi
Platform Tools	toolsOk
Operating System	Red Hat Enterprise Linux 5 (64-bit)
CPU Affinity	
Snapshots	1
Advanced Settings	0
Resources	Available
Management Engine GUID	bdff0558-8828-11e2-9abc-005056af00b4

- **Lifecycle** shows the date of discovery and the last analysis. If a retirement date and time or owner has been set, these display as well.



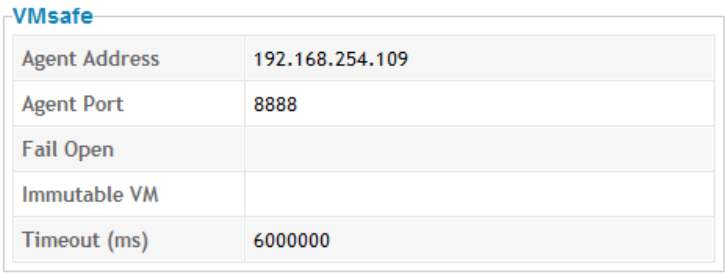
Lifecycle	
Discovered	Fri Mar 08 19:45:34 UTC 2013
Last Analyzed	Tue Mar 12 15:06:37 UTC 2013
Retirement Date	Never
Owner	Administrator
Group	EvmGroup-administrator

- **Relationships** include information on the parent host, genealogy such as parent and child virtual machines, and drift.

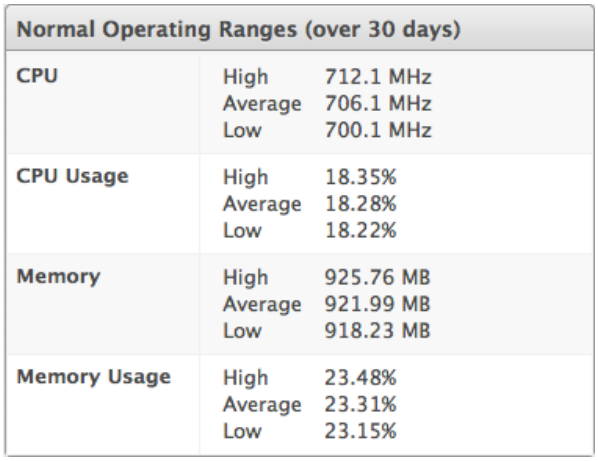


Relationships	
Management System	Virtual Center (192.168.252.14)
Cluster	Testing-Production Cluster
Host	vi4esxm2.manageiq.com
Resource Pool	RP3
Datastores	StarM1-Testing1
Service	None
Parent VM	None
Genealogy	Show parent and child VMs
Drift History	1
Analysis History	1

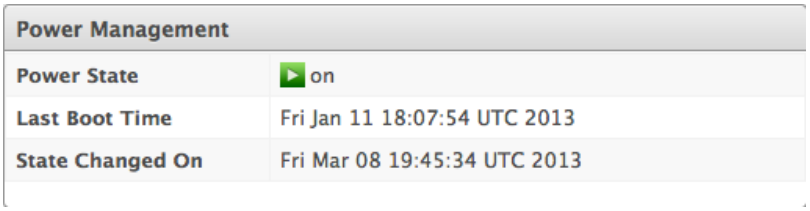
- **Storage Relationships** shows relationships to Filers, LUNs, Volumes and File Shares.
- **VMsafe** shows properties of the VMsafe agent if it is enabled.



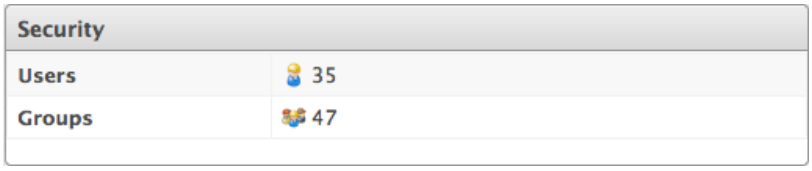
- **Normal Operating Ranges** shows the values for the normal operating range for this virtual machine. These statistics are used in calculating right sizing recommendations.



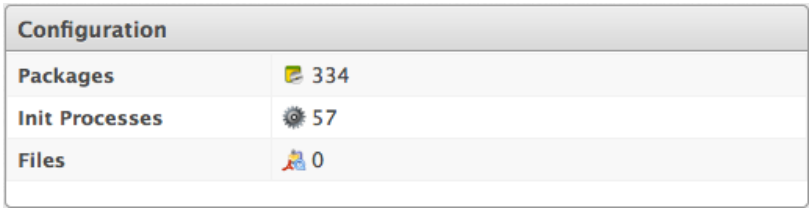
- **Power Management** displays the current power state, last boot time, and last power state change. **State Changed On** is the date that the virtual machine last changed its power state. This is a container view of the power state, therefore a restart of the operating system does not cause the container power state to change and will not update this value.



- **Security** includes information on users, groups, and security patches. Recall that the items shown on the **Summary** screen change based on the guest operating system.



- **Configuration** includes information on applications, services, packages, and init processes. This section changes depending on the base operating system.



- **Datastore Allocation Summary** shows how many and how much disk space has been allocated to this virtual machine as well as disk alignment and thin provisioning information.
- **Datastore Actual Usage Summary** shows how much disk and memory the virtual machine is actually using.

Datastore Actual Usage Summary	
Disks	42 GB
Memory	4 GB
Snapshots	0 Bytes
Total Datastore Used Space	42.03 GB
Unused/Overcommited Allocation	(31.99 MB) * Overallocated

- **Diagnostics** provides a link to viewing running processes and the information from the latest collected event logs.
- **Smart Management** shows all tags assigned to this virtual machine.

Viewing the Operating System Properties

View details of the operating system from the **Virtual Machine Summary** or the accordion. For Windows systems, see **Account Policies** for the virtual machine.

1. From **Compute > Infrastructure > Virtual Machines**, click on the item to view its **Summary**.
2. From the **Properties** section, click **Operating System**.

An expanded view of the operating systems properties and **Account Policies** displays. This varies based on the operating system.

Viewing Virtual Machine or Template Snapshot Information

View the list of snapshots to see a history of their creation and size. Infrastructure Automation provides the description, size, and creation time of the snapshot as well as a view of the genealogy of the snapshots.

Note:

Snapshot size is only available after the successful completion of a **SmartState Analysis**.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the appropriate accordion containing the item you wish to view the snapshots of.
3. Click on the item to view its **Summary**.
4. From the **Summary**, click *Snapshots* in the **Properties** area.
5. The list of snapshots show in a tree format and captures their genealogy.

Viewing User Information for a Virtual Machine or Template

Infrastructure Automation's **SmartState Analysis** feature returns user information. Drill into the user to get details on the user's account, including group memberships.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the item to view user information.
3. Click on the item to view its **Summary**.
4. From the **Security** section of the **Virtual Machine Summary**, click **Users**.
5. Click the user to view details.

Viewing Group Information for a Virtual Machine or Template

Infrastructure Automation's **SmartState Analysis** feature returns group information. Explore a group to get a list of its users.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the item to view user information.
3. Click on the item to view its **Summary**.
4. From the **Security** section of the **Virtual Machine Summary**, click **Groups**.
5. Click the group to view users.

Viewing Genealogy of a Virtual Machine or Template

Infrastructure Automation detects the lineage of a virtual machine. View a virtual machine's lineage and compare the virtual machines that are part of its tree. This also allows tagging of virtual machines that share genealogy.

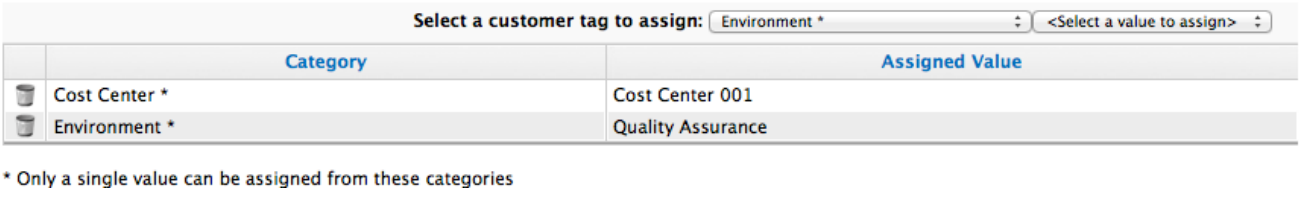
1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the item to view genealogy.
3. Click on the item to view its **Summary**.
4. From the **Relationships** area in the **Summary**, click **Genealogy**.

Comparing Genealogy of a Virtual Machine or Template

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the item to view genealogy.
3. Click on the item to view its **Summary**.
4. From the **Relationships** area in the **Summary**, click **Genealogy**.
5. Check the items to compare.
6. Click (**Compare Selected VMs**).

Tagging Virtual Machines or Templates with a Common Genealogy

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the item to view genealogy.
3. Check the items to tag.
4. Click (**Policy**), and then (**Edit Tags**).
5. Select a customer tag from the first dropdown, and then a value for the tag.



Detecting Drift on Virtual Machines or Templates

The configuration of a virtual machine might change over time. **Drift** is the comparison of a virtual machine to itself at different points in time. The virtual machine needs analysis at least twice to collect this information. Detecting drift provides you the following benefits:

- See the difference between the last known state of a machine and its current state.
- Review the configuration changes that happen to a particular virtual machine between multiple points in time.
- Review the host and datastore association changes that happen to a particular virtual machine between multiple points in time.
- Review the classification changes that happen to a virtual machine between two time checks.
- Capture the configuration drifts for a single virtual machine across a time period.

Detect drift on virtual machines or templates:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click on the item to view its **Summary**.
3. From the **Relationships** area in the **Summary**, click **Drift History**.
4. Check the analyses to compare.
5. Click (**Select up to 10 timestamps for Drift Analysis**) at the top of the screen. The results display.
6. Check the **Drift** sections on the left to view in your comparison.
7. Click **Apply**.
8. The following descriptions pertain to the **Expanded View** . Whether you see the value of a property or an icon representing the property depends on the properties type.
    - A property displayed in the same color as the base means the compared analysis matches the base for that property.
    - A property displayed in a different color from the base means the compared analysis does not match the base for that property.
9. If you are in the **Compressed View** , the values of the properties are not displayed. All items are described by the icons shown below.
    - A (**checkmark**) means that the compared analysis matches the base for that property. If you hover over it, the value of the property will display.
    - A (**triangle**) means the compared analysis does not match the base for that property. If you hover over it, the value of the property displays. Click the minus sign next to the sections name to collapse it.
10. To limit the scope of the view, you have three buttons in the **Resource** button area.
    - Click (**All attributes**) to see all attributes of the sections you selected.
    - Click (**Attributes with different values**) to see only the attributes that are different across the drifts.
    - Click (**Attributes with the same values**) to see only the attributes that are the same across drifts.
11. To limit the mode of the view, there are two buttons in the **Resource** button area.
    - Click (**Details Mode**) to see all details for an attribute.
    - Click (**Exists Mode**) to only see if an attribute exists compared to the base or not. This only applies to attributes that can have a Boolean property. For example, a user account exists or does not exist, or a piece of hardware that does or does not exist.

This creates a drift analysis. Download the data or create a report from your drift for analysis using external tools.

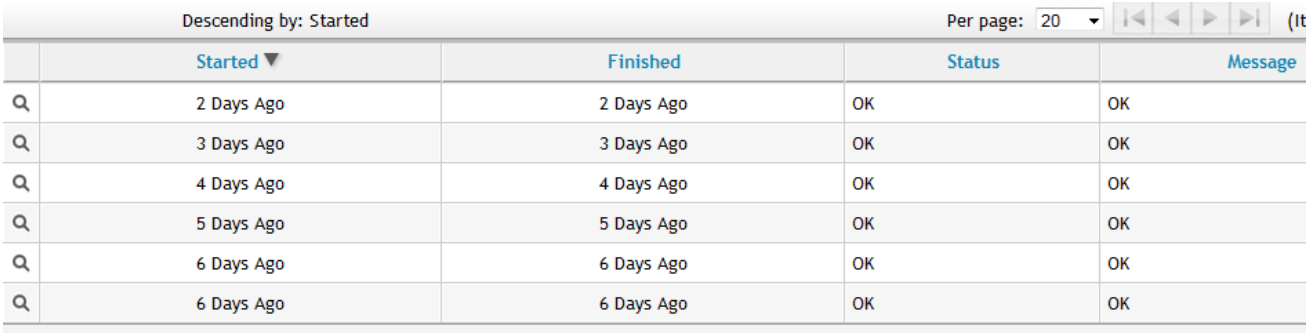
Creating a Drift Report for a Virtual Machine or Template

1. Create the comparison to analyze.
2. Click (**Download**).
3. Click the output button for the type of report you want.
   - Click (**Download drift report in text format**) for a text file.
   - Click (**Download drift report in CSV format**) for a csv file.
   - Click (**Download drift report in PDF format**) for a PDF file.

Viewing Analysis History for a Virtual Machine or Template

Each time a SmartState Analysis is performed on a virtual machine, a record is created of the task. This information is accessed either from the virtual machine accordion or the virtual machine summary. Use this detail to find when the last analysis was completed and if it completed successfully. If the analysis resulted in an error, the error is shown here.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the item to view genealogy.
3. Click on the item to view its **Summary**.
4. From the **Relationships** area in the **Summary**, click **Analysis History**. A history of up to the last 10 analyses is displayed.



5. Click on a specific analysis to see its details.

Viewing Disk Information for a Virtual Machine or Template

Each time a SmartState Analysis is performed on a virtual machine or template, information on the disks associated with the item is collected. This includes free and used space information as well as the type of disk and file system.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click on the item to view its **Summary**.
3. From **Datastore Allocation Summary**, click **Disks**.

A list of the disks for the item with type, file system, size, and usage information is displayed.

Reconfiguring Virtual Machines

Memory, processors, disks, ISOs, and CD/DVD Drives can be reconfigured on existing VMware and Red Hat Virtualization virtual machines using the **Reconfigure this VM** button.

You can reconfigure multiple components on a virtual machine using one request, or make each reconfiguration separately. Confirm your changes using the **Submit** button.

The following restrictions apply to adding and removing Red Hat Virtualization virtual machine disks: - Supported by Red Hat Virtualization 4.0 and above. - Only a single bootable disk is supported. - The virtual machine requires at least one existing disk to allow adding additional disks. This is because the destination storage cannot be specified from the Infrastructure Automation dialog, so the storage associated with the existing disk is reused.

Adding a Disk to a Virtual Machine

A disk can be added to a VMware or Red Hat Virtualization virtual machine with the following steps:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**, and then (**Reconfigure this VM**).
4. Click **Add Disk**.
5. Specify the disk type, mode, size, and dependency options.
6. Click **Submit**.

After the disk has been added, you can view the new disk by navigating to **Compute > Infrastructure > Virtual Machines**. . Open the target virtual machine, then click **Devices** to view the new disk.

Removing a Virtual Machine Disk

A disk can be removed from a VMware or Red Hat Virtualization virtual machine with the following steps:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**, and then (**Reconfigure this VM**).
4. Click **Delete** next to the disk to remove.
5. Click **Submit**.

Resizing a Virtual Machine Disk

Note:

This functionality is available on VMware virtual machines only.

You can extend a VMware virtual machine's disk with the following steps:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**), and then (**Reconfigure this VM**).
4. Click **Resize** next to the disk you want to resize to show resizing options.
5. Specify the desired memory size and units (MB or GB). A disk can only be extended in size.
6. Click **Confirm Resize** to select the values. Alternatively, selecting **Cancel Resize** shows the original values.
7. Click **Submit**.

The disk resizing request is then processed for the virtual machine.

Increasing or Decreasing a Virtual Machine's Memory Size

You can increase or decrease a VMware or Red Hat Virtualization virtual machine's memory size with the following steps:

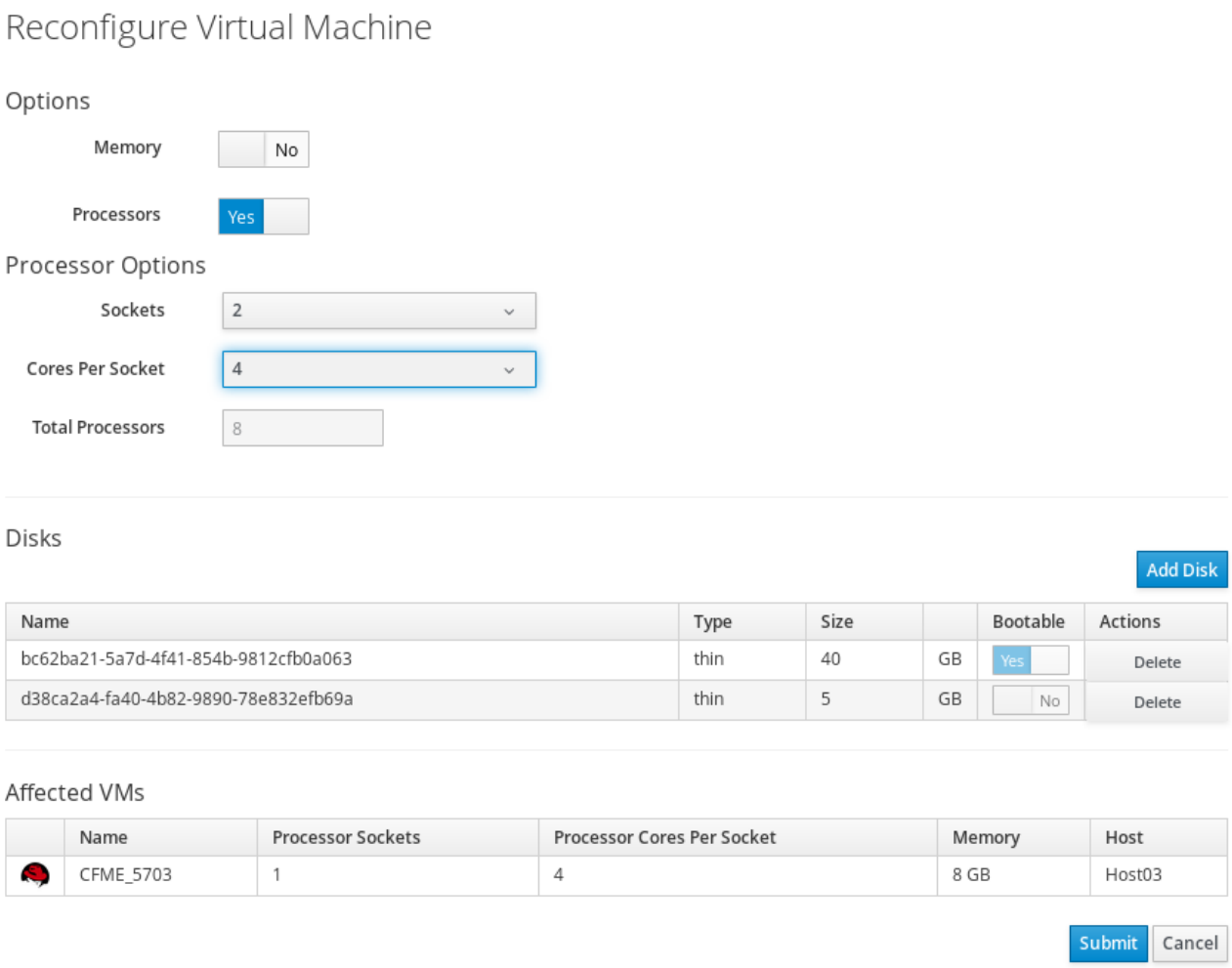
1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**, and then (**Reconfigure this VM**).
4. Select **Yes** next to **Memory** to show memory options.
5. Specify the desired memory size and units (MB or GB).
6. Click **Submit**.

The memory add request is then processed for the virtual machine.

Reconfigure a Virtual Machine's Processors

You can reconfigure a VMware or Red Hat Virtualization virtual machine's processors with the following steps:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**, and then **Reconfigure this VM**.
4. Select **Yes** next to **Processors** to show processor options.



5. Specify the number of sockets and the number of cores per socket. The reconfiguration screen will calculate the **Total Processors**.
6. Click **Submit**.

The virtual machine's processors are then reconfigured.

Adding or Removing Virtual Machine Network Adapters

Note:

This functionality is available on VMware virtual machines only.

You can add or remove network adapters on a VMware virtual machine from the **Reconfigure this VM** button.

Adding a Network Adapter to a VMware Virtual Machine

To add a network adapter to a virtual machine:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**, and then (**Reconfigure this VM**).
4. Click **Add Network** next to the disk you want to resize to show resizing options.
5. Select the type of adapter from the list under **vLan**.
6. Click **Confirm Add**.

7. Click **Submit**.

A request to add the network adapter is then processed for the virtual machine. You can view the details for the new network adapter by navigating to the **Reconfigure this VM** area. When the network adapter is added, a name and MAC address are assigned to the adapter.

Removing a Network Adapter from a VMware Virtual Machine

To remove a network adapter from a virtual machine:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**, and then (**Reconfigure this VM**).
4. Click **Delete** next to the network adapter you want to remove.
5. Click **Submit**.

Attach or Remove an ISO (VMware Virtual Machines Only)

You can attach an ISO to a VMware virtual machine using following steps:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**, and then (**Reconfigure this VM**).
4. Under **CD/DVD Drives**, click **Connect**.
5. Select the **Host File** from the list to attach and click **Confirm Connect**.
6. Click **Submit**.

The ISO is now attached to the VMware virtual machine.

You can remove an ISO on VMware virtual machine using following steps:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine.
3. Click **Configuration**, and then (**Reconfigure this VM**).
4. Under **CD/DVD Drives**, locate the **Host File** and click **Disconnect**.
5. Click **Confirm**.
6. Click **Submit**.

The ISO is removed from the VMware virtual machine.

Viewing Event Logs for a Virtual Machine or Template

Using an **Analysis Profile**, collect event log information from your virtual machines.

Note:

This feature is only available for Windows.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click the accordion for the item to view event logs.
3. Click on the item to view its **Summary**.

4. From **Diagnostics** click **Event Logs**.

The collected event log entries are displayed. Sort this list by clicking on the column headers.

Remote Consoles

A console is a graphical window that allows you to view the start up screen, shut down screen, and desktop of a virtual machine, and to interact with that virtual machine in a similar way to a physical machine.

Infrastructure Automation offers the following support for HTML5-based VNC, SPICE, and WebMKS consoles:

- VNC and SPICE consoles for Red Hat Virtualization Manager with websocket proxy.
- VMware's WebMKS HTML5-based console type.
- VNC consoles for OpenStack using OpenStack-supplied websocket proxy.

Note:

- VMware no longer supports the MKS console type. Also, VMRC is no longer a browser plugin but a native desktop application. As a result, the VMware MKS and the VMRC plugin options have been disabled in Infrastructure Automation.
- Currently, attempting to connect to the VMware WebMKS console for a virtual machine fails when the server security type is set to **2** for that virtual machine.
- Due to VMware licensing restrictions, Red Hat cannot ship the WebMKS SDK in Infrastructure Automation. For information on how to configure WebMKS support in Infrastructure Automation, see Configuring WebMKS Support in Infrastructure Automation.

All of the above make use of the websocket protocol supported by all recent versions of browsers, and can use SSL to encrypt the websocket connection.

- OpenStack Infrastructure Automation only makes an API call to get the URL for the console and open that console in a web browser; see Directly Connect to a VNC Console in the Red Hat OpenStack Platform *Instances and Images Guide* for more details.
- Red Hat Enterprise Virtualization Manager and VMware By default, the websocket connection runs over HTTPS or HTTP based on how the application was accessed. Under an appliance, you will most likely use HTTPS, and, therefore, the websocket connection will be wss:// (websocket with SSL).

  When configuring Red Hat Virtualization Manager for virtual machine console access, set the display type for each virtual machine to **noVNC** or **SPICE HTML5**. For more information on configuring console options, see Configuring Console Options in the Red Hat Virtualization *Virtual Machine Management Guide*.

Configuring Console Access to VMware ESXi Hosts At A Network Layer

When configuring access to the VNC or HTML5 console, make sure that at a network layer:

- All VNC ports (5900-6000) are opened from the machine on which you access the Infrastructure Automation Console to the Infrastructure Automation.
- All VNC ports (5900-6000) are opened from the Infrastructure Automation to each VMware ESXi host running virtual machines that you want to access.
- The firewall on VMware ESXi hosts is enabled and that the VMware ESXi host firewall ports are opened.
- The VNC service (**gdbserver**) is running and that the **gdbserver** service has an association with ports 5900-6000 usually defined with a **/etc/vmware/firewall/service.xml** firewall rules configuration.

  The **gdbserver** ruleset must be enabled on each ESXi host running virtual machines that will be accessed through the HTML5 console or VNC console on the Infrastructure Automation. The ruleset can be configured on the host itself, or using the VMware vCenter web user interface.

The following procedures apply to VMware vCenter 5.0 and later.

Using SSH to Configure VMware ESXi Hosts to Enable Console Access

Configure the **gdbserver** ruleset on the host using SSH.

1. Access the host:

```
# ssh host@example.com
```

2. Set the **gdbserver** parameter:

   ```
   # esxcli network firewall ruleset set --ruleset-id gdbserver --enabled true
   ```

3. Confirm that the ruleset is active:

   ```
   # esxcli network firewall ruleset list
   ```

Using the VMware vCenter Web Interface to Configure ESXi Hosts to Enable Console Access

Configure the **gdbserver** ruleset on the host using the VMware vCenter web user interface.

1. Select the ESXi host in the VMware vCenter web interface.
2. Click the **Manage** tab.
3. Click the **Settings** sub tab.
4. Click **System > Security Profile** from the list on the left.
5. Click **Edit**.
6. Select the **gdbserver** ruleset, and then click **OK**.

Configuring the VMware ESXi Host Firewall Ports for Console Access

Follow these steps to configure the VMware ESXi host firewall ports for HTML5 or VNC console access to guest virtual machine consoles. The firewall ports must be enabled on each VMware ESXi host running virtual machines that will be accessed through the HTML5 or VNC console on the Infrastructure Automation.

1. Log in to your vSphere Client and select **Home > Inventory > Hosts and Clusters**.
2. In the **Hosts/Clusters** tree view, select the VMware ESXi host you want to configure for HTML5 or VNC console access.
3. Select the **Configuration** tab and open the **Software** box.
4. Select **Security Profile**.
5. Browse to the **Firewall Properties** dialog window by selecting the **Properties** link from the **Firewall** section.
6. In the **Firewall Properties**, scroll down to **GDB Server** and select it.
7. Click **OK**.

Configuring WebMKS Support in Infrastructure Automation

Complete the following steps to enable WebMKS support in Infrastructure Automation.

1. Log in to the Infrastructure Automation user interface appliance console as the root user.
2. On the Infrastructure Automation user interface appliances, create a folder titled **webmks** in the **/var/www/miq/vmdb/public/** directory.

   ```
   /var/www/miq/vmdb/public/webmks
   ```

3. Download and extract the contents of VMware WebMKS SDK into the **webmks** folder.

Opening a Console for a Virtual Machine

Open a web-based VNC or SPICE console for a virtual machine.

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click on the virtual machine that you want to access.
3. Click (**Access**) and select **VM Console** or **Web Console**.

The virtual machine console opens in a new tab in your browser.

Managing Virtual Machine Snapshots

A snapshot is a view of a virtual machine's operating system and applications on any or all available disks at a given point in time. Take a snapshot of a virtual machine before you make a change to it that may have unintended consequences. You can use a snapshot to return a virtual machine to a previous state.

View virtual machines by infrastructure providers by navigating to **Compute > Infrastructure > Providers**.

Creating a Virtual Machine Snapshot

To create a snapshot of a virtual machine:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine, and open it to view its details.
3. Under **Properties**, click **Snapshots** to show information about the virtual machine's snapshots.
4. Click (**Create a new snapshot for this VM**).
5. Enter snapshot details in **Description**.
6. (Optional) Select **Snapshot VM memory** for the snapshot to preserve the virtual machine's current runtime state. This option will appear only if the VM **Power State** is (**On**).
7. Click **Create**.

Deleting a Virtual Machine Snapshot

To delete a virtual machine snapshot:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the target virtual machine, and open it to view its details.
3. Under **Properties**, click **Snapshots** to show information about the virtual machine's snapshots.
4. Select the snapshot to delete under **Available Snapshots**.
5. Click and select **Delete Selected Snapshot**.
6. Click **OK**.

Reverting a Virtual Machine to a Snapshot

A virtual machine can be reverted to a previous state using a snapshot.

The virtual machine state must be powered `off` for the **Revert to selected snapshot** option to be available.
To revert a virtual machine to a snapshot:

1. Power the virtual machine off.
2. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
3. Select the target virtual machine, and open it to view its details.
4. Under **Properties**, click **Snapshots** to show information about the virtual machine's snapshots.
5. Select the snapshot to revert to under **Available Snapshots**.
6. Click (**Revert to selected snapshot**).
7. Click **OK**.

Creating a Template Based on a Virtual Machine

A template is a copy of a virtual machine (VM) that you can use to simplify the subsequent, repeated creation of similar VMs. Templates capture the configuration of software, configuration of hardware, and the software installed on the VM on which the template is based.

Note:

Virtual machines must meet the following criteria to publish as templates:

- The VM must not be blank, archived or orphaned.
- The VM must be associated with a oVirt / RHV provider.
- oVirt / RHV provider 4.0 or higher, with `use_ovirt_engine_sdk` set to `true` in `/var/www/miq/vmdb/config/settings.yml` on your Infrastructure Automation appliance.
- The VM power state is `OFF`.
- Sealing a template is not supported for Windows OS VMs.

To create a template based on an existing VM:

1. Browse to menu: **Compute > Infrastructure > Providers**.
2. Click on a oVirt / RHV provider.
3. Select a VM.
4. Click (**Lifecycle**), then click (**Publish selected VM to a Template**).
5. Under the **Request** tab, provide **Request Information** and identify a **Manager**:
    1. Enter an **E-Mail** address.
    2. Provide a **First** and **Last** name in the individual fields.
    3. Enter a **Manager** name.
6. Assign tags to the new template in the **Purpose** tab.
    1. Click on a tag category and check applicable tags.
7. Use fields in the **Catalog** tab to identify the source VM and provide details for the template:
    1. Select the source VM from the **Name** field.
    2. (Optional) Check **Seal Template**.

       Note:

       Sealing, which uses the `virt-sysprep` command, removes system-specific details from a virtual machine before creating a template based on that VM. This prevents the original VM's details from appearing in subsequent VMs that are created using the same template. It also ensures the functionality of other features, such as predictable vNIC order.
    3. The **Number of VMs** indicates the output will be a single template.
    4. Provide a **Name** and **Description** for the created template.
8. Under the **Environment** tab, specify information for the storage domain in which to create the template's disks.
    1. Check **Choose Automatically** to allow Infrastructure Automation to determine the destination cluster and datastore.

       Or, manually enter cluster and datastore information using the following steps:
    2. Select the cluster with which to associate the template from the **Cluster** list. By default, this is identical to the cluster of the source VM.
    3. Choose a **Datastore** for the destination to create the template's disks.
9. Provide provisioning and retirement information for VMs based on the template in the **Schedule** tab.
    1. Select **Schedule** or **Immediately on Approval** for **Schedule Info** to determine when to provision the VM.
    2. Set the **Time to Retirement** under **Lifespan** using the drop-down menu.

10. Click **Submit**.

Retiring Virtual Machines

Specify a retirement date and time for virtual machines or instances using Infrastructure Automation.

To set retirement date and time:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the virtual machines or instances for retirement.
3. Click (**Lifecycle**), then select **Set Retirement Date**.
4. Under **Enter Retirement Date as**, select **Specific Date and Time**.
5. Select a date using the calendar control, then click (**Select Time**).
6. Specify the hour and minute for retirement.
7. Select a **Retirement Warning** from the list.
8. Click **Save**.

Migrating Virtual Machines

Infrastructure Automation supports migrating virtual machines between datacenters, clusters, resource pools, folders and hosts. Create and submit a request with detailed environment information to which to migrate the VM.

To create a request to migrate a virtual machine:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Click **VMs & Templates** on the accordion menu.
3. Select the VM to migrate.
4. Click **Configuration**, and then **Lifecycle**.
5. Click **Migrate selected items**. The Migrate Virtual Machine form will appear.
6. Under the **Request** tab:
    1. Provide an **E-mail** address, **First Name** and **Last Name**.
    2. Include any **Notes** and a Manager's **Name**.
7. Under the **Environment** tab:
    1. Select a **Datacenter** to migrate the VM to.
    2. Select a **Cluster** from the list.
    3. Choose a **Resource Pool**.
    4. Select a **Folder** to migrate the VM to.
    5. Under **Host** choose a **Filter** and click on a **Name** from the list that appears.
    6. **Filter** the **Datastore** and click a **Name** from the results.
8. Under the **Schedule** tab:
    1. Select **Schedule Info** for migrating the virtual machine. Provide a **Provision on** date if selecting **Schedule**.
9. Click **Submit**.

Resource Pools

Resource pools are used to allocate CPU and memory across a group of virtual machines.

Removing a Resource Pool

If a resource pool is decommissioned or requires troubleshooting, it might require removal from the VMDB.

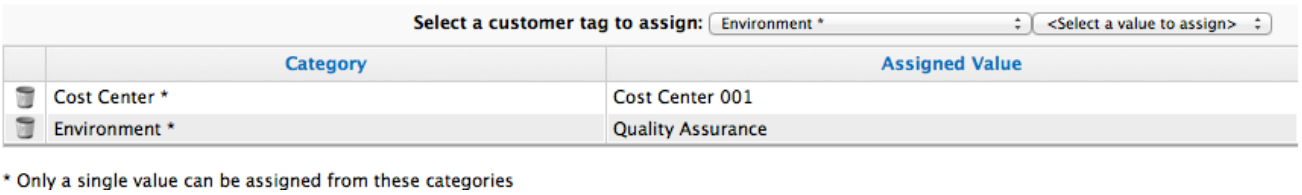
1. Browse to menu: **Compute > Infrastructure > Resource Pools**.
2. Click on the resource pool to remove.
3. Click **Configuration**, and then (**Remove from the VMDB**).
4. Click **OK**.

The resource pool is removed. The virtual machines remain in the VMDB, but are no longer associated with this resource pool.

Tagging a Resource Pool

Use tags to categorize a resource pool.

1. Browse to menu: **Compute > Infrastructure > Resource Pools**.
2. Click the resource pool to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. Select a customer tag from the first dropdown, and then a value for the tag.



Viewing the Resource Pool Summary

Use the Resource Pool Summary to see the number of discovered virtual machines, the parent host, and the parent cluster. It is the default view when you click on one resource pool.

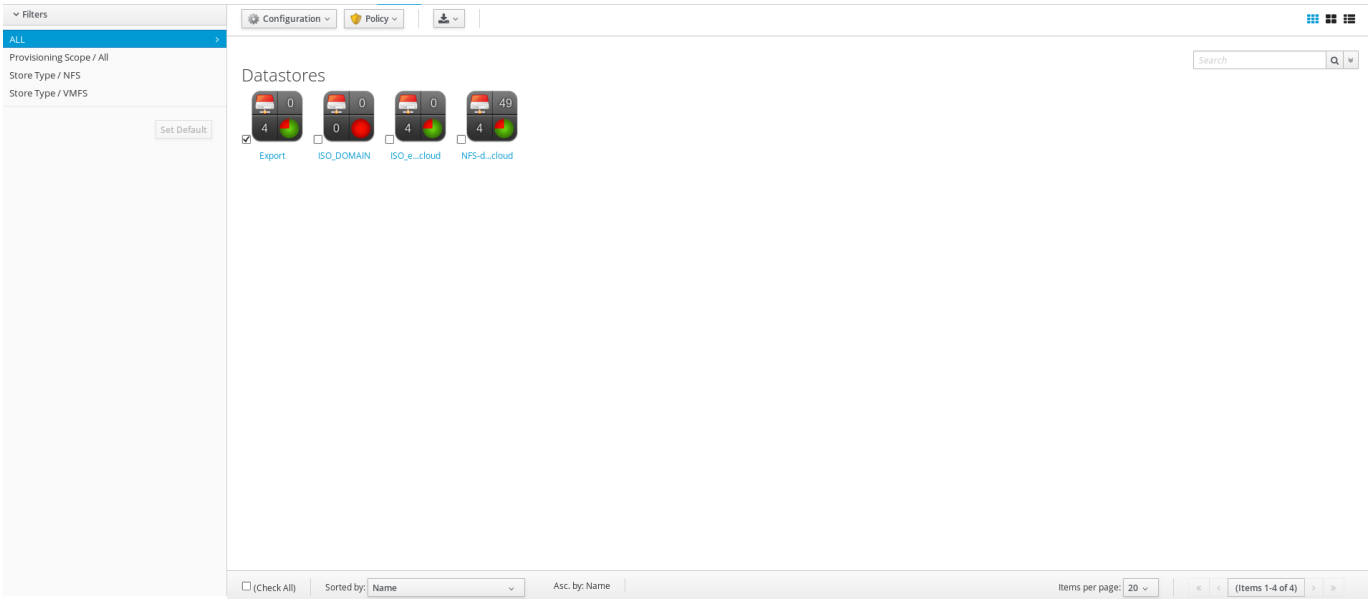
Resource Pools Accordion

Use the **Resource Pools** accordion to access the properties and objects associated with the resource pool.

- Click **Properties** to view the Resource Pools summary screen.
- Click **Relationships** to see the clusters, virtual machines, and hosts related to this resource pool.

Datastores

A storage location is considered a device where digital information resides and is connected to a resource. Infrastructure Automation detects, analyzes, and collects capacity and utilization data for both VMFS and NFS datastores. Datastores connected to a provider are automatically created on discovery. On creation of a repository, a datastore is automatically created.



After detecting datastores, you might want to examine them more closely to see virtual machines, hosts, and available space.

1. File system type
2. Number of hosts
3. Number of virtual machines
4. Available space

Performing SmartState Analysis on Datastores

Analyze a datastore to collect information on the types of files on a datastore, and to see the number of managed/registered, managed/unregistered, and unmanaged virtual machines. To perform a SmartState analysis, the datastore is accessible from a running host and valid security credentials are supplied for that host.

Note:

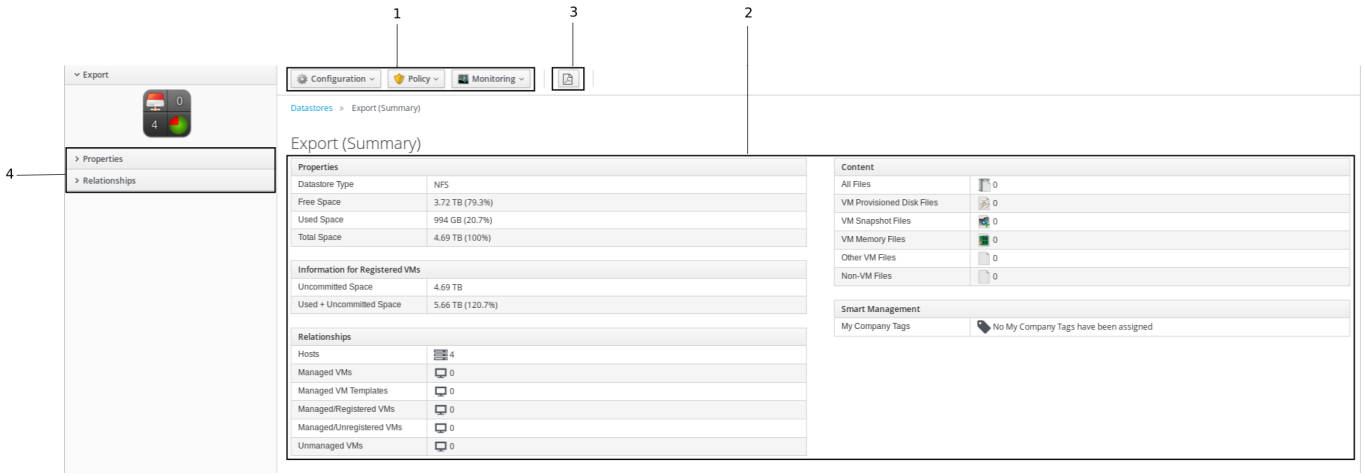
- SmartState analysis on datastores is processed by the **Provider Operations** role. It is enabled by default.
- Be aware that executing a SmartState analysis on a datastore from the console takes a while to return data on the content. If capacity and utilization roles are enabled, Infrastructure Automation performs the analysis automatically on a scheduled basis approximately every 24 hours.

1. Browse to menu: **Compute > Infrastructure > Datastores**.
2. Select the datastores to analyze.
3. Click **Configuration**, and then (**Perform SmartState Analysis**).
4. Click **OK**.

Viewing a Datastore

You can click on a specific datastore to view its details. The screen provides you with a datastore taskbar, virtual thumbnail, accordion, and summary.

Datastore Management Screen.



1. **Datastore Taskbar**: Choose between **Configuration**, **Policy** and **Monitoring** options for the selected datastore.
2. **Datastore Summary**: See summary such as datastore properties, storage, VM information.
3. **Datastore PDF**: Generates datastore summary in PDF format.
4. **Datastore Accordion**: See details about **Properties**, **Relationships**, **Storage Relationships** and **Content** for the chosen datastore.

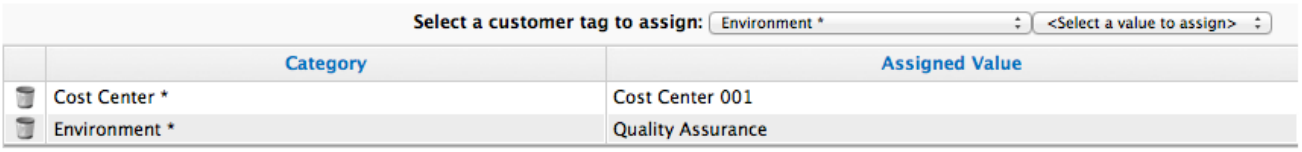
Note:

To view **Content** section details, run a SmartState Analysis on the datastore. For information on how to perform SmartState Analysis, see Performing SmartState Analysis on Datastores.

Tagging a Datastore

Use tags to categorize a datastore.

1. Browse to menu: **Compute > Infrastructure > Datastores**.
2. Click the datastore to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. Select a customer tag from the first list, and then a value for the tag from the second list.



5. Select more tags as required.
6. Click **Save**.

Viewing Capacity and Utilization Charts for a Datastore

You can view capacity and utilization data for a datastore.

Note:

Infrastructure Automation requires network visibility to your provider assigned the server role of *Capacity & Utilization Collector* to enable this feature.

1. Browse to menu: **Compute > Infrastructure > Datastores**, then click the Datastore for which to view Capacity and Utilization data.
2. Click (**Monitoring**), and then (**Utilization**).
3. From **Interval**, select to view hourly or daily data points and the dates to view data.
4. Use **Show VM Types** to include only managed/registered, managed/unregistered, or unmanaged virtual machines.
    - **Managed/Registered VM** - A virtual machine connected to a host and exists in the VMDB. Also, a template connected to a management system and exists in the VMDB.

      Note:

      Templates cannot be connected to a host.
    - **Managed/Unregistered VM** - A virtual machine or template that resides on a repository that is no longer connected to a management system or host, but exists in the VMDB. A virtual machine previously considered registered might become unregistered if the virtual machine is removed from management system inventory.
    - **Not Managed** - Files discovered on a datastore that do not have a virtual machine associated with them in the VMDB. These files might be registered to a management system that Infrastructure Automation does not have configuration information. Possible causes might be the management system has not been discovered or the management system has been discovered but no security credentials are provided.
5. Use **Time Profiles** to select a time range for the data.

Note:

Daily charts only include full days of data. If a day does not include all the 24 data points for a day, the data does not show for that day.

For information about data optimization including utilization trend reports, see Data Optimization.

Removing a Datastore

If a datastore no longer contains any files associated with the virtual environment, remove it from the VMDB. This button is enabled only if a datastore is completely empty.

1. Browse to menu: **Compute > Infrastructure > Datastores**.
2. Click on the Datastore to remove.
3. Click **Configuration**, and then (**Remove Datastore from Inventory**).
4. Click **OK**.

Data Optimization

Infrastructure Automation optimization functions allow you to view utilization trends in your environment. In addition, you can predict where you have capacity for more virtual machines, see Planning Where to Put a New Virtual Machine in the *Deployment Planning Guide*.

Note: To access the utilization report features, you will first need to enable data collection in Infrastructure Automation; see the following sections in the *Deployment Planning Guide*:

- Assigning the Capacity and Utilization Server Roles
- Capacity and Utilization Data Collected

Utilization Trends

Infrastructure Automation allows you to view the resource utilization of your clusters, providers, and datastores. You can choose from summary, details, or report view.

Viewing a Utilization Trend Summary

This procedure shows you how to view a utilization trend summary.

1. Browse to menu: **Overview > Utilization**.
2. Click **Summary** if it is not already selected.
3. Expand the tree on the left side until you can see the wanted providers, clusters, or datastores.
4. Click the item.
5. Use the **Options** section in the **Summary** tab to change the characteristics of the data.
    - Use **Trends** to select how far back you want to calculate the trend.
    - Use **Selected Day** for the date you want to see percent utilization for in the chart on the **Summary** tab.
    - Use **Classification** to see trends only for a specific applied tag.
    - Use **Time Profile** to select an existing time profile. If the logged on user does not have any time profiles available, this option will not show.
    - Select a **Time Zone**.

Viewing Detail Lines of a Utilization Trend

This procedure shows you how to view detail lines of a utilization trend.

1. Browse to menu: **Overview > Utilization**.
2. Expand the tree on the left side until you can see the wanted providers, clusters, or datastores.
3. Click the item.
4. Click **Details** if it is not already selected.
5. From the **Options** area, select how far back you want to view the trends for and any classifications you want to use.

Viewing a Report of a Utilization Trend

To find out more about resource utilization, view utilization trend reports.

1. Browse to menu: **Overview < Utilization**.
2. Expand the tree on the left side until you can see the wanted providers, clusters, or datastores.
3. Click the item.
4. Click **Report** if it is not already selected.
5. From the **Options** area, select how far back you want to view the trends for and any classifications you want to use.

PXE Servers

PXE servers are used by Infrastructure Automation to bootstrap virtual machines for the purpose of provisioning. They include images for different operating systems that can be customized using customization templates and are used in conjunction with IPMI Servers.

Availability Zones

An availability zone is a provider-specific method of grouping cloud instances and services. Infrastructure Automation uses Amazon EC2 regions and OpenStack Nova zones as availability zones.

Viewing an Availability Zone

You can click on an availability zone to view its details. The screen provides you with an availability zone accordion and an availability zone summary page.

- You can choose between graphical or text view of the datastore summary.
- Use the availability zone accordion to view the **Properties** of the zone and its **Relationships** to other cloud resources.
- Use the availability zone summary to see details on **Relationships** (**Cloud Provider**, **Instances**) and **Smart Management** (**Company Tags**).

Viewing Availability Zone Relationships

Use the availability zone accordion's **Relationship** section to see items related to an availability zone.

1. Browse to menu: **Compute > Clouds > Availability Zones**.
2. Click the availability zone to view the configuration.
3. From the availability zone accordion, click **Relationships**.
4. Click the type of resource relationship to view as a list.

Cloud Tenants

A tenant is an OpenStack term for an organizational unit or project. Infrastructure Automation creates cloud tenants to match existing OpenStack tenants for managing resources and controlling visibility of objects.

Note:

OpenStack tenant mapping must be enabled for cloud tenants to be created. See Tenant Mapping for details.

OpenStack uses tenants for the following reasons:

- Assigning users to a project
- Defining quotas for a project
- Applying access and security rules for a project
- Managing resources and instances for a project

This helps administrators and users organize their OpenStack environment and define limits for different groups of people. For example, one project might require higher quotas and another project might require restricted access to certain ports. OpenStack allows you to define these limits and apply them to a project.

Infrastructure Automation can abstract information from tenants including quotas and relationships to other OpenStack objects.

To see multiple tenants in Infrastructure Automation, the user authenticating to your OpenStack environment from Infrastructure Automation must be configured to have visibility into these tenants.

Note:

This section describes OpenStack cloud tenant usage. For information about tenants created in Infrastructure Automation for access and resource control, see Access Control in *General Configuration*.

Tenant Mapping

When adding an OpenStack cloud or infrastructure provider, enable *tenant mapping* in Infrastructure Automation to map any existing tenants from that provider.

This means Infrastructure Automation will create new cloud tenants to match each existing OpenStack tenant; each new cloud tenant and its corresponding OpenStack tenant will have identical resource assignments (including user and role synchronization) with the exception of quotas. Tenant quotas are not synchronized between Infrastructure Automation and OpenStack, and are available for reporting purposes only. You can manage quotas in Infrastructure Automation but this will not affect the quotas created in OpenStack.

During a provider refresh, Infrastructure Automation will also check for any changes to the tenant list in OpenStack. Infrastructure Automation will create new cloud tenants to match any new tenants, and delete any cloud tenants whose corresponding OpenStack tenants no longer exist. Infrastructure Automation will also replicate any changes to OpenStack tenants to their corresponding cloud tenants.

If you leave tenant mapping disabled, Infrastructure Automation will not create cloud tenants or tenant object hierarchy from OpenStack.

Viewing Cloud Tenant Relationships

From **Compute > Clouds > Tenants**, click on a specific cloud tenant to view its details.

The screen provides you with a cloud tenant accordion and a cloud tenant summary.

- Use cloud tenant summary views to change how you are looking at the **Summary**.
- Use the cloud tenant accordion to view the **Properties** of the tenant and its **Relationships**.
- Use the cloud tenant summary to see details on **Relationships** (**Cloud Provider**, **Security Groups**, **Instances**, and **Images**), **Quotas** (including all OpenStack **Compute**, **Network**, and **Volume** quotas) and **Smart Management** (**Company Tags**).

Volumes

A volume is a block storage device that you can attach to or detach from an instance to manage the storage available to that instance. Volumes are managed through storage managers, which are added automatically to Infrastructure Automation when the corresponding provider is added.

Amazon Elastic Block Store Manager Volumes

This section outlines the actions that you can perform on Amazon Elastic Block Store manager volumes.

Creating Volumes

You can create and attach volumes to your storage manager.

To create a volume:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Click **Configuration**, then click (**Add a new Cloud Volume**).
3. Select the Amazon Elastic Block Store manager from the **Storage Manager** list.
4. Select an availability zone from the **Availability Zone** list.
5. Enter a **Volume Name**.
6. Select the type of the volume from the **Cloud Volume Type** list.

   Note:

   See Amazon EBS Volume Types for more information on volume types.
7. Enter the size of the volume in gigabytes (GB).
8. Select whether the volume should be encrypted using the **Encryption** toggle.
9. Click **Add**.

The volume appears in the list of volumes after it has been provisioned.

Creating a Snapshot of a Volume

You can create a snapshot of a volume to preserve the state of the volume at a specific point in time. The snapshot can be used to create a duplicate of the volume.

To create a snapshot of a volume:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Click the volume to snapshot to open the volume's summary page.

3. Click **Configuration**, then click **Create a Snapshot of this Cloud Volume**.
4. Enter a name for the snapshot in **Snapshot Name**.
5. Click **Save**.

Click **Cloud Volume Snapshots** on the summary page of a volume to view the snapshots for that volume.

Attaching a Volume to an Instance

To attach a volume to an instance:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to attach.
3. Click **Configuration**, then click **Attach selected Cloud Volume to an Instance** to open the **Attach Cloud Volume** screen.
4. Select an instance from the list.
5. Optionally, enter a **Device Mountpoint**.
6. Click **Attach**.

Detaching a Volume from an Instance

To detach a volume from an instance:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to detach.
3. Click **Configuration**, then click **Detach selected Cloud Volume from an Instance** to open the **Detach Cloud Volume** screen.
4. Select an instance from the list.
5. Click **Detach**.

Editing a Volume

You can edit several properties of existing volumes.

To edit a volume:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to edit to open its summary page.
3. Click **Configuration**, then click (**Edit this Cloud Volume**).
4. Enter a new **Volume Name**.
5. Select a new volume type from the **Cloud Volume Type** list.

   Note:

   See Amazon EBS Volume Types for more information on each of the volume types.
6. Enter a new size in gigabytes.
7. Click **Save**.

Deleting a Volume

To delete a volume from a storage manager:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to delete.
3. Click **Configuration**, then click (**Delete selected Cloud Volumes**).

OpenStack Block Storage Manager Volumes

This section outlines the actions that you can perform on OpenStack Block Storage manager volumes.

Creating Volumes

You can create and attach volumes to your storage manager.

To create a volume:

After creating a volume, only the volume name can be edited.

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Click **Configuration**, then click (**Add a new Cloud Volume**).
3. Select the OpenStack Block Storage manager from the **Storage Manager** list.
4. Enter a **Volume Name**.
5. Enter the size of the volume in gigabytes (GB).
6. Under **Placement**, select the cloud tenant to attach it to.
7. Click **Add**.

The volume appears in the list of volumes after it has been provisioned.

Creating a Backup of a Volume

You can create a backup of a volume to protect against data loss, and restore it in the future.

For OpenStack Block Storage managers, the `openstack-cinder-backup` service must be enabled on the OpenStack Block Storage manager to create a volume backup.
To create a backup of a volume:

1. Browse to menu: **Storage < Block Storage < Volumes**.
2. Click the volume you want to back up to open the volume's summary page.
3. Click **Configuration**, then click (**Create a Backup of this Cloud Volume**).
4. Enter a name for the backup in **Backup Name**.
5. (Optional) Select **Incremental?** to take an incremental backup of the volume instead of a full backup.

   Note:

   You can take an incremental backup of a volume if you have at least one existing full backup of the volume. An incremental volume saves resources by capturing only changes made to the volume since its last backup. See Create an Incremental Volume Backup in the *Storage Guide* for more information.
6. (Optional) Select **Force?** to allow backup of a volume attached to an instance.

Note:

Selecting the **Force** option will back up the volume whether its status is *available* or *in-use*. Backing up an *in-use* volume ensures data is crash-consistent.

7. Click **Save**.

View a volume's backups by clicking **Cloud Volume Backups** on the volume's summary page.

Note:

See Back Up and Restore a Volume in the *Storage Guide* for more information about backups.

Restoring a Volume from a Backup

In case of data loss, you can restore a volume from a backup with the following steps:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Click the volume whose backup you want to restore. This will open the volume's summary page.
3. Click **Configuration**, then click (**Restore from a Backup of this Cloud Volume**).
4. Select the volume to restore from in the **Cloud Volume Backup** list.
5. Click **Save**.

Restoring a Cloud Volume from a Backup

In case of data loss, you can restore from a cloud volume backup with the following steps:

1. Browse to menu: **Storage > Block Storage > Volume Backups**.
2. Select a **Cloud Volume Backup** to restore.
3. Click **Configuration**, then click (**Restore backup to Cloud Volume**).
4. Select the **Volume** to restore from the backup.
5. Click **Save**.

Deleting a Cloud Volume Backup

Delete unnecessary cloud volume backups through the following steps:

1. Browse to menu: **Storage > Block Storage > Volume Backups**.
2. Select the **Cloud Volume Backups** to delete.
3. Click **Configuration**, then click (**Delete selected Backups**).
4. Click **OK** to confirm your choice.

Creating a Snapshot of a Volume

You can create a snapshot of a volume to preserve the state of the volume at a specific point in time. The snapshot can be used to create a duplicate of the volume.

To create a snapshot of a volume:

1. Browse to menu: **Storage > Block Storage > Volumes**.

2. Click the volume to snapshot to open the volume's summary page.
3. Click **Configuration**, then click (**Create a Snapshot of this Cloud Volume**).
4. Enter a name for the snapshot in **Snapshot Name**.
5. Click **Save**.

Click **Cloud Volume Snapshots** on the summary page of a volume to view the snapshots for that volume.

Note:

See Create, Use, or Delete Volume Snapshots in the *Storage Guide* for more information about snapshots.

Attaching a Volume to an Instance

To attach a volume to an instance:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to attach.
3. Click **Configuration**, then click (**Attach selected Cloud Volume to an Instance**) to open the **Attach Cloud Volume** screen.
4. Select an instance from the list.
5. Optionally, enter a **Device Mountpoint**.
6. Click **Attach**.

Detaching a Volume from an Instance

To detach a volume from an instance:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to detach.
3. Click **Configuration**, then click **Detach selected Cloud Volume from an Instance** to open the **Detach Cloud Volume** screen.
4. Select an instance from the list.
5. Click **Detach**.

Editing a Volume

Only the volume name can be edited on an existing volume.

To edit a volume's name:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to edit to open its summary page.
3. Click **Configuration**, then click (**Edit this Cloud Volume**).
4. Enter the new **Volume Name**.
5. Click **Save**.

Deleting a Volume

To delete a volume from a storage manager:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to delete.
3. Click **Configuration**, then click (**Delete selected Cloud Volumes**).

IBM Power Systems Virtual Servers Block Storage Manager Volumes

This section outlines the actions that you can perform on IBM Power Systems Virtual Servers Block Storage manager volumes.

Creating Volumes

You can create and attach volumes to your storage manager.

To create a volume:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Click **Configuration**, then click **Add a new Cloud Volume**.
3. Select the desired IBM Power Systems Virtual Servers instance from the **Storage Manager** list.
4. Enter a **Volume Name**.
5. Select the type of the volume from the **Cloud Volume Type** list.

   Note:

   See Power Systems Virtual Servers documentation for more information on volume types.
6. Enter the size of the volume in gigabytes (GB).
7. Click **Add**.

The volume appears in the list of volumes after it has been provisioned.

Attaching a Volume to an Instance

To attach a volume to an instance:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to attach.
3. Click **Configuration**, then click (**Attach selected Cloud Volume to an Instance**) to open the **Attach Cloud Volume** screen.
4. Select an instance from the list.

   Note:

   Note: A VM cannot have disks from different storage types. Only VMs matching storage type will be available for selection.
5. The **Device Mountpoint** field is not used.
6. Click **Attach**.

Detaching a Volume from an Instance

To detach a volume from an instance:

1. Browse to menu: **Storage > Block Storage > Volumes**.

2. Select the volume to detach.
3. Click **Configuration**, then click **Detach selected Cloud Volume from an Instance** to open the **Detach Cloud Volume** screen.
4. Select an instance from the list.
5. Click **Detach**.

Deleting a Volume

To delete a volume from a storage manager:

1. Browse to menu: **Storage > Block Storage > Volumes**.
2. Select the volume to delete.
3. Click **Configuration**, then click (**Delete selected Cloud Volumes**).

Flavors

Flavors indicate the resource profiles available for instances. Each flavor contains a value set for CPUs, CPU cores and memory. Flavors allow you to pre-configure resource settings, which you can then apply during instance provisioning. You can also change the flavor of a provisioned instance; see Resizing an Instance for instructions.

Infrastructure Automation provides the ability to view individual flavor information and instances currently using the flavor.

Creating a Flavor

You can create a new flavor for the provider.

1. Browse to menu: **Compute > Clouds > Flavors**.
2. Click **Configuration**, then click (**Add a new Flavor**).
3. Select the provider from the **Provider** list.
4. Enter a **Name** for the flavor.
5. Enter **RAM size in MB**.
6. Enter **VCPUs**.
7. Enter **Disk size in GB**.
8. Enter **Swap size in MB**.
9. Enter **RXTX factor**. This is an optional property allows servers with a different bandwidth to be created with the RXTX factor. The default value is 1. That is, the new bandwidth will be the same as that of the attached network.
10. Click **Public** to set `True` or `False`. The default is `True`. If you set it to false, select cloud tenants from the **Cloud Tenant** list.
11. Click **Add**.

Viewing a Flavor

You can click on a specific flavor to view its details. The screen provides you with a flavor accordion and a flavor summary.

- Use flavor summary views to change how you are looking at the summary.
- Use the flavor accordion to view the **Properties** of the flavor and its **Relationships**.

- Use the flavor summary to see details on **Properties** (**CPUs**, **CPU Cores**, **Memory**), **Relationships** (**Cloud Provider**, **Instances**), and **Smart Management** (**Company Tags**).

Viewing Flavor Relationships

Use the **Relationship** section in the flavor accordion to see items related to the flavor.

1. Browse to menu: **Compute > Clouds > Flavors**.
2. Click a flavor to view the configuration.
3. From the flavor accordion, click **Relationships**.
4. Click the type of resource to see the flavor's relationships.

Deleting a Flavor

1. Browse to menu: **Compute > Clouds > Flavors**.
2. Select the flavors you want to remove from the list.
3. Click **Configuration**, then click **Remove selected Flavors**.
4. Cick **OK** on the warning window to remove selected flavors permanently.

Cloud Networks

Infrastructure Automation enables configuration and administration for the software-defined networking component of Red Hat OpenStack Platform. The virtual network infrastructure enables connectivity between instances and the physical external network.

This section describes common cloud network administration tasks, such as adding and removing subnets and routers to suit your Red Hat OpenStack Platform providers.

Creating and Administering Cloud Networks

Create a network to provide instances a place to communicate internally and receive IP addresses using Dynamic Host Configuration Protocol (DHCP). A network can also be integrated with external networks in your Red Hat OpenStack Platform deployment or elsewhere, such as the physical network.

Note:

- Keystone API v3 is required to create cloud tenants on Red Hat OpenStack Platform cloud providers. For more information, see OpenStack Identity (keystone) in the Red Hat OpenStack Platform *Architecture Guide*.
- For further details on cloud network objects and administration, see the *Networking Guide* in the Red Hat OpenStack Platform documentation..

Adding a Cloud Network

Add a new cloud network following the steps in this procedure:

1. Browse to menu: **Network > Networks**.
2. Click **Configuration** and then click **Add a new Cloud Network**.
3. In the **Network Providers** area, select a **Network Manager** from the drop-down menu.
4. Under **Placement**, select a **Cloud Tenant**.
5. In the **Network Provider Information** section, choose a **Provider Network Type**.
    - If **Local** is selected, provide a **Physical Network** name.
    - If **GRE** is selected, provide a **Physical Network** name and **Segmentation ID**.

6. In the **Network Information** area:
    1. Provide a descriptive **Network Name** based on the role the network will perform.
    2. Enable an **External Router**
    3. Set the **Administrative State** to control whether the network is immediately available.
    4. Establish **Shared** status of the network.
7. Click **Add**.

Editing Cloud Network Details

To edit network information details of a cloud network:

1. Browse to menu: **Networks > Networks**.
2. Select a network from the list view.
3. Click **Configuration**, then (**Edit selected Cloud Network**).
4. Edit **Network Information** fields.
5. Click **Save**.

Deleting a Cloud Network

To delete a cloud network:

1. Browse to menu: **Networks > Networks**.
2. Select a cloud network from the list view.
3. Click **Configuration**, then click **Delete selected Cloud Networks**.

Creating and Administering Subnets

Infrastructure Automation enables creation and administration of subnets for your cloud networks. Create subnets in pre-existing networks as means to grant network connectivity to instances. Subnets are the means by which instances are granted network connectivity. Each instance is assigned to a subnet as part of the instance creation process.

Consider proper placement of instances to best accommodate their connectivity requirements:

- Tenant networks in OpenStack Networking can host multiple subnets.
- Subnets are isolated from one another.
- Host distinctly different systems on different subnets within the same network.
- Instances on one subnet that wish to communicate with another subnet must have traffic directed by a router.
- Place systems requiring a high volume of traffic amongst themselves in the same subnet, avoiding routing and subsequent latency and load issues.

Adding a Subnet

Add a subnet to an existing cloud network following the procedure below.

1. Browse to menu: **Networks > Subnets**.
2. Click **Configuration**, then click **Add a new Cloud Subnet**.
3. Select a **Network Manager**.
4. Under **Placement**, select a **Cloud Tenant**.
5. Provide the following **Cloud Subnet** details:
    1. A descriptive **Subnet Name**.
    2. The **Gateway** IP address of the router interface for the default gateway.
    3. **Enable DHCP** services for the subnet. DHCP allows automated distribution of IP settings to instances.

4. Select the **IP Version**. The IP address range in the Network Address field must match whichever version you select.
5. Input the **Subnet CIDR** address in CIDR format, which contains the IP address range and subnet mask in one value.

   Note:

   Determine the address by calculating the number of bits masked in the subnet mask and append that value to the IP address range. For example, the subnet mask 255.255.255.0 has 24 masked bits. To use this mask with the IPv4 address range 192.168.122.0, specify the address 192.168.122.0/24.

Editing a Cloud Subnet

To edit the details of a cloud subnet:

1. Browse to menu: **Networks > Subnets**.
2. Click on a subnet from the list view.
3. Click **Configuration**, then **Edit this Cloud Subnet**.
4. Edit **Cloud Subnet details** fields.
5. Click **Save**.

Configuring Network Routers

Infrastructure Automation enables configuration for Red Hat OpenStack Platform provider cloud network routing services using an SDN-based virtual router.

- Routers are a requirement for your instances to communicate with external subnets, including those out in the physical network.
- Routers and subnets connect using interfaces, with each subnet requiring its own interface to the router.
- A router's default gateway defines the next hop for any traffic received by the router.
- A router's network is typically configured to route traffic to the external physical network using a virtual bridge.

Adding a Network Router

Add a network router to an existing cloud network by following the procedure below.

1. Browse to menu: **Networks > Network Routers**.
2. Click **Configuration** and click **Add a new Router**.
3. In the **Network Provider** area, select a **Network Manager**.
4. Provide a **Router Name**.
5. Under **External Gateway**:
   1. If **Yes** is selected, provide the following:
      1. Choose to **Enable Source NAT**. In Source Network Address Translation (SNAT), he NAT router modifies the IP address of the sender in IP packets. SNAT is commonly used to enable hosts with private addresses to communicate with servers on the public Internet.
      2. Select a **Network**.
      3. Select a **Subnet**.
6. Select a **Cloud Tenant**.
7. Click **Add**.

Editing Network Router Details

To edit the details of a network router:

1. Browse to menu: **Networks > Network Routers**.
2. Select a network router from the list view.
3. Click **Configuration**, then **Edit selected Router**.

4. Edit required fields.
5. Click **Save**.

Adding an Interface to a Network Router

Interfaces allow you to interconnect routers with subnets. As a result, the router can direct any traffic that instances send to destinations outside of their intermediate subnet.

To add an interface to a network router:

1. Browse to menu: **Networks > Network Routers**.
2. Select a network router from the list view.
3. Click **Configuration**, then (**Add Interface to this Router**).
4. Select a **Subnet** from the list.
5. Click **Add**.

Removing a Network Router Interface

You can remove an interface to a subnet if you no longer require the router to direct its traffic.

To remove an interface:

1. Browse to menu: **Networks > Network Routers**.
2. Select a network router in the list view.
3. Click **Configuration**, then (**Remove Interface from this Router**).
4. Confirm your choice.

Deleting a Network Router

To delete a network router:

1. Browse to menu: **Networks > Network Routers**.
2. Select a network router from the list view.
3. Click **Configuration**, then click **Delete selected Routers**.

Creating Floating IPs

Floating IP addresses allow you to direct ingress network traffic to your cloud network instances. Define a pool of validly routable external IP addresses, which can then be dynamically assigned to an instance. All incoming traffic destined for that floating IP is routed to the instance to which it has been assigned.

Note:

Red Hat OpenStack Networking allocates floating IP addresses to all projects (tenants) from the same IP ranges/CIDRs. As a result, every subnet of floating IPs is consumable by any and all projects. Manage this behavior using quotas for specific projects.

Adding Floating IPs.

Floating IP addresses allow you to direct ingress network traffic to your instances.

Add floating IP addresses to an existing cloud network by following the procedure below.

1. Browse to menu: **Networks > Floating IPs**.
2. Click **Configuration** and click Add a new Floating IP.
3. Select a **Network Manager**.
4. Choose an **External Network**
5. Under **Association Information** provide the following:
    1. (Optional) An **Associated Port** for the floating IP.
    2. (Optional) The **Floating IP Address**.

6. Select a **Cloud Tenant**.
7. Click **Add**.

Managing Port Association of a Floating IP

To manage the port associations of a floating IP:

1. Browse to menu: **Networks > Floating IPs**.
2. Click on a floating IP from the list view.
3. Click **Configuration**, then **Manage the port association of this Floating IP**.
4. To associate a port, add a new **Port id**
5. To disassociate a port, delete the **Port id** field information.
6. Click **Save**.

Deleting a Floating IP

To delete a floating IP

1. Browse to menu: **Networks > Floating IPs**.
2. SClick on a floating IP from the list view to view its summary page.
3. Click **Configuration**, then click **Delete this Floating IP**.

Security Groups

You can group instances using security groups to restrict port or IP address accessibility. Security groups can be created and assigned to instances using Infrastructure Automation instance provisioning.

Cloud providers that currently support this function include: Amazon EC2, OpenStack, and Red Hat Enterprise Virtualization.

Editing Security Group Details

Editing security group information allows users to make changes to existing security group details and firewall rules, in additional to adding new firewall rules.

To edit details of a security group:

1. Browse to menu: **Networks > Security Groups**.
2. Click on a security group to view the summary page.
3. Click **Configuration**, then **Edit this Security Group**.
4. Under **Security Group Information**, edit the **Security Group Name** and the **Security Group Description**.
5. Edit existing **Firewall Rules** or add new firewall rules by clicking **Add a Firewall Rule**.
6. Click **Save**.

Viewing Security Groups

This procedure describes how to view security groups.

1. Browse to menu: **Networks > Security Groups**.
2. Click the desired security groups for viewing the details.
    - In **Properties**, you can view the basic information of the security group.
    - In **Relationships**, you can view the cloud provider and the instances associated with the security group.
    - In **Firewall Rules**, you can view a list of ports and IP ranges that are accessible.

      Note:

      This box is not available if you have not set any rules for your security group.

Tagging Security Groups

Apply tags to security groups to categorize them.

1. Browse to menu: **Networks > Security Groups**.
2. Select the security group to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. Select a customer tag to assign from the dropdown menu.
5. Select a value to assign.
6. Click **Save**.

Deleting a Security Group

To delete a security group:

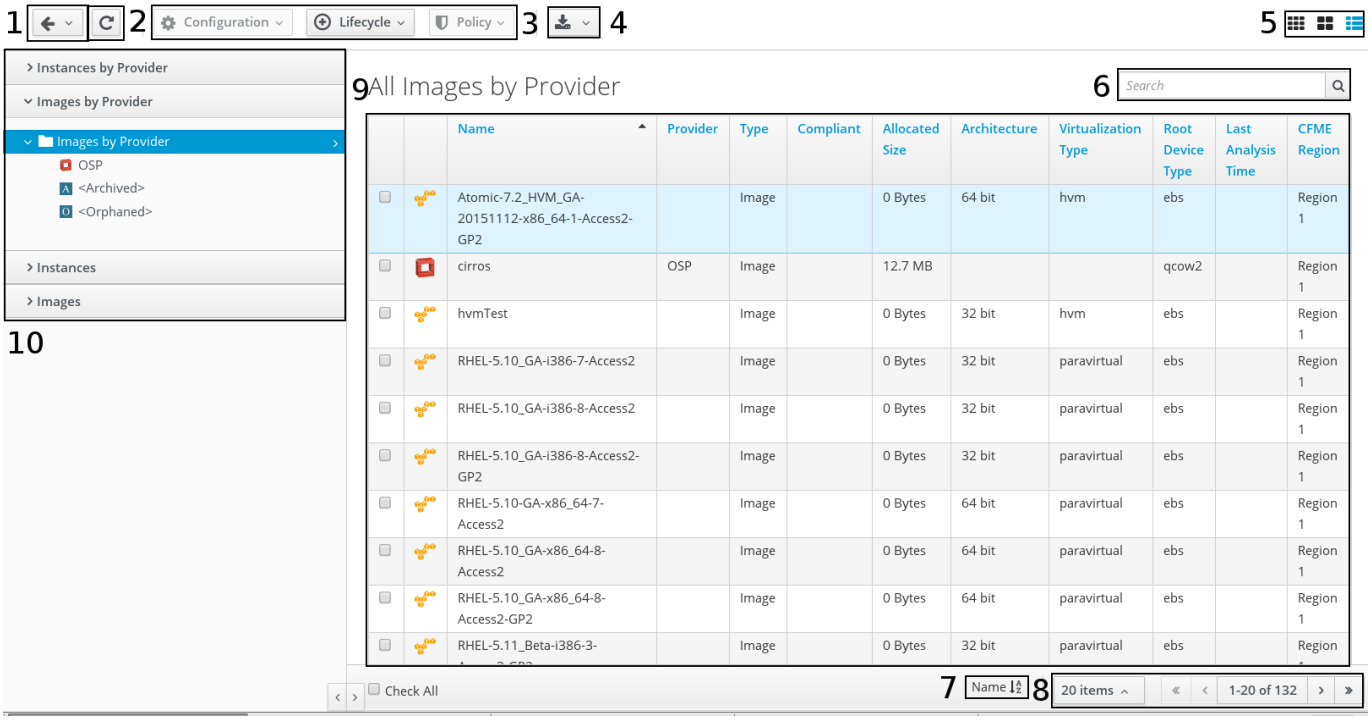
1. Browse to menu: **Networks > Security Groups**.
2. Click on a security group in the list view to view its summary page.
3. Click on **Configuration**, then click **Delete this Security Group**.

Instances and Images

Images are the static templates containing the software configuration from which you provision a running **Instance** - a virtual machine, with which you can interact - on your cloud provider.

The **Instance** and **Images** containers, combined with the ability to analyze information inside each instance or image, provides in-depth information across the cloud environment. This rich set of information enables Infrastructure Automation users to improve problem resolution times and effectively manage instances and images in their cloud environment.

The **Instances** and **Images** pages display all instances and images the server discovered from your cloud providers. The taskbar on each page is a menu driven set of buttons that provide access to functions related to instances and images.



1. History button
2. Refresh screen button
3. Taskbar
4. Download buttons

5. View buttons
6. Name search bar/Advanced Search button
7. Sort dropdown
8. Navigation bar
9. Main area in List View
10. Cloud/Filter Navigation

Console uses Virtual Thumbnails to describe instances and images. Each thumbnail contains four quadrants by default. This allows you to glance at an instance or image for a quick view of its contents.

1. Top left quadrant: Operating system of the Instance
2. Bottom left quadrant: Instance Cloud Provider
3. Top right quadrant: Power state of Instance or Status icon
4. Bottom right quadrant: Number of Snapshots for this Instance

Icon	Description
	Template: Cloud Image
	Retired: Instance has been retired
	Archived: Instance has no provider or availability zone associated with it.
	Orphaned: Instance has no availability zone but does have a provider associated with it.
	Disconnected: Instance is disconnected.
	On: Instance is powered on.
	Off: Instance is powered off.
	Suspended: Instance has been suspended.

The **Instances** page has four accordions organizing your instances and images in different ways. All of these accordions share a set of common controls:

- Use **Instances by Provider** and **Images by Provider** to view your instances and images organized by provider. In addition, you can see archived and orphaned items here.
- Use the **Instances** to view, apply filters, and collect information about all of your instances.
- Use **Images** to view, apply filters, and collect information about all of your images.

Through the console, you can view your instances and images in multiple ways:

- Filter instances
- Change views
- Sort
- Create a report
- Search by Tags
- Search by collected data

Filtering Instances and Images

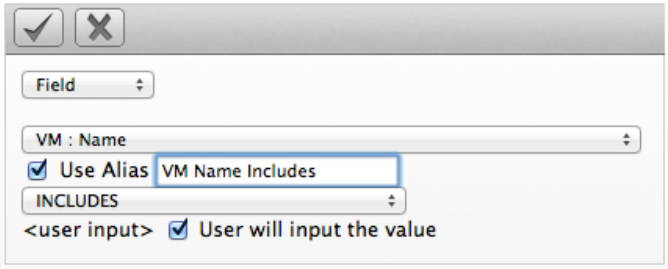
The **Instance Filter** accordion is provided so that you can easily navigate through groups of instances. You can use the ones provided or create your own through **Advanced Filtering** capabilities.

Using an Instance or Image Filter

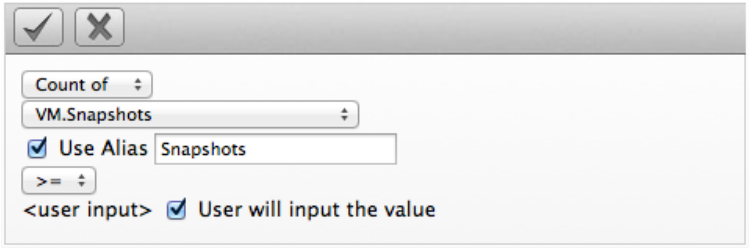
1. Browse to menu: **Compute > Clouds > Instances**.
2. Click on the **Instances** or **Images** accordion.
3. Click on the desired filter from the left pane.

Creating an Instance or Image Filter

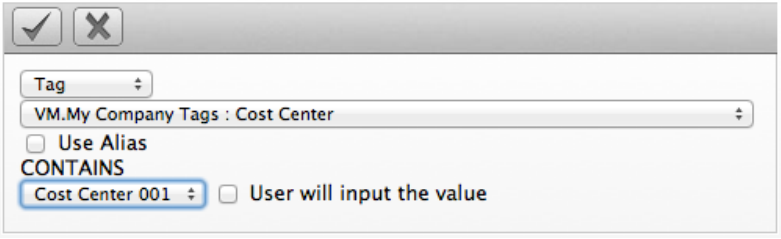
1. Browse to menu: **Compute > Clouds > Instances**.
2. Go to the **Instances** or **Images** accordion.
3. Click **All Instances** or **All Images**, then click (**Advanced Search**) to open the expression editor.
4. Use the expression editor to choose the appropriate options for your criteria. Based on what you choose, different options will show.
   - For all of the types of searches, you have the options of creating an alias and requested user input. Select **Use Alias** to create a user friendly name for the search. If you are requested user input for the search, this text will show in the dialog box where the input is requested.
   - Click **Field** to create criteria based on field values.



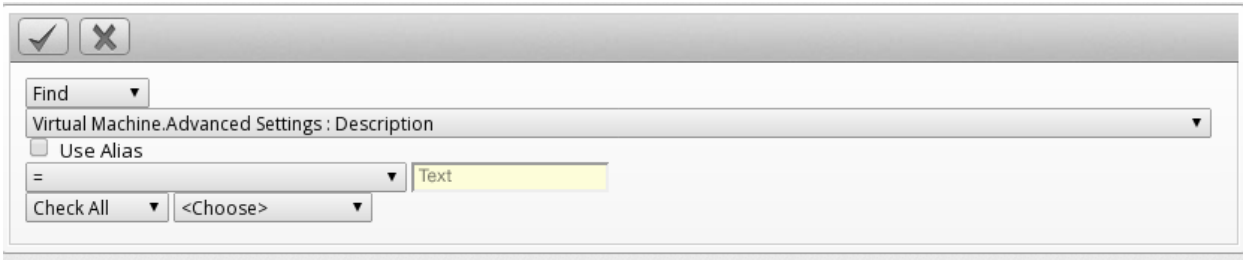
   - Click **Count of** to create criteria based on the count of something, such as the number of snapshots for an instance, or the number of instances on a host.



   - Click **Tag** to create criteria based on tags assigned to your virtual infrastructure, such as for power states or production tagging.



   - Click **Find** to seek a particular value, and then check a property.



5. Click (**Commit Expression Element Changes**) to add the expression.
6. Click **Save**.
7. Type in a name for the search expression in **Save this Instance search as**. To set the filter to show globally, check **Global Search**.
8. Click **Save**.

The filter is saved and shows in the **My Filters** area of the **Filter** accordion. If you checked **Global Search**, the filter shows there.

Loading a Report Filter or Search Expression

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the items to search (either **Instances** or **Images**).
3. Click (**Advanced Search**) to open the expression editor.
4. Click **Load**.
5. Select either a saved instance search or an instance report filter.

   Note:

   The set of items to select will depend on the type of resource you are searching.
6. Click **Load** to load the search expression.
7. If you want to edit the expression, click on it and make any edits for the current expression.
   - Click (**Commit expression element changes**) to add the changes.
   - Click (**Undo the previous change**) to remove the change you just made.
   - Click (**Redo the previous change**) to put the change that you just made back.
   - Click AND (**AND with a new expression element**) to create a logical AND with a new expression element.
   - Click OR (**OR with a new expression element**) to create a logical OR with a new expression element.
   - Click NOT (**Wrap this expression element with a NOT**) to create a logical NOT on an expression element or to exclude all the items that match the expression.
   - Click (**Remove this expression element**) to take out the current expression element.
8. Click **Load**.
9. Click **Apply**.

Changing Views for Instances and Images

While you can set the default view for different pages from the settings menu, then menu: **Configuration > My Settings > Default Views**, the current view can also be controlled from the Instances pages.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the items to view.
3. Click the appropriate button for the desired view.
    - Click for **Grid View**.
    - Click for **Tile View**.
    - Click for **List View**.

Sorting Instances and Images

Virtual machines and images can be sorted by Name, Availability Zone, Flavor, Cloud Provider, Compliant, Last Analysis Time, and Region.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the desired items to sort.
3. To sort instances or images when in grid or tile view:
    - From the **Sort by dropdown**, click the attribute to sort.
4. To sort instances or images when in list view:
    - Select the **List View**.
    - Click on the **Column Name** to sort. For example, click on **Availability Zone** to sort by the name of the availability zone.

Creating an Instance or Image Report

For a listing of instances and images, you can create a quick report in CSV, TXT, or PDF formats.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the desired items for report creation.
3. Click (**Download**).
    - Click for a **TXT** file.
    - Click for a **CSV** file.
    - Click for a **PDF** file.

Searching for Instances or Images

To the right of the taskbar on the **Instances** page, you can enter names or parts of names for searching. You can search in the following ways.

- Type characters that are **included** in the name. For example, if you type **`sp1`**, all Instances whose names include **`sp1`** appear, such as **`Windows2003`** and **`Sp1clone`**.
- Use **`*`** at the end of a term to search for names that begin with specific characters. For example, type **`v*`** to find all instances whose names begin with the letter **`v`**.
- Use **`*`** at the beginning of a term to search for names that end with specific characters. For example, type **`*\sp2`** to find all instances whose names end with **`sp2`**.
- Erase all characters from the search box to go back to viewing all instances.

Search for instances and images:

1. Browse to menu: **Compute > Clouds > Instances**.

2. Click the accordion for the desired items to search.

3. In the **Name Filter** bar in the upper right corner of the window, type your criteria.
4. Click (**Search by Name within results**) or press **Enter**.
5. Type in other criteria to filter on what is currently displayed.
6. Click (**Search by Name within results**) or press **Enter**.

Analyzing Instances and Images with SmartState Analysis

Analyze an instance to collect metadata such as user accounts, applications, software patches, and other internal information. If Infrastructure Automation is not set up for automatic analysis, perform a manual analysis of an instance. To perform a SmartState Analysis, Infrastructure Automation requires a running SmartProxy with visibility to the instance's storage location so that a snapshot can be created.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the items to analyze.
3. Check the instances and images to analyze.
4. Click **Configuration**, and then **Perform SmartState Analysis** on the taskbar.
5. Click **OK**.

Note:

Restrictions on Displaying Files Collected

- File size bigger than 20k characters
- File with missing name
- Non MIME .conf file, with non ascii characters
- Non MIME .conf file, without content
- MIME .exe binary file
- MIME .txt non binary file
- Non MIME .conf ascii file

SmartState Analysis for instances runs as a process independent from providers. For example, a successful SmartState Analysis of a host does not mean SmartState Analysis for instances will be successful. Ensure to enter credentials for the provider that contains the instance for the SmartState Analysis to work.

Comparing Instances and Images

You can compare multiple instances in Infrastructure Automation server. This allows you to see how different instances are from their original image. This helps detect missing patches, unmanaged user accounts, or unauthorized services.

Use the comparison feature to:

- Compare multiple instances from different hosts
- Compare multiple instances side-by-side
- Quickly see similarities and differences among multiple instances and a base
- Narrow the comparison display to categories of properties
- Print or export in the comparison results to a PDF or CSV file

Compare instances and images:

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the items to analyze.
3. Click the checkboxes for the items to compare.
4. Click **Configuration**, and then (**Compare Selected items**). The comparison displays in a compressed view with a limited set of properties listed.
5. To delete an item from the comparison, click (**Remove this VM from the comparison**) at the bottom of the items column.
6. To view many items on one screen, go to a compressed view by clicking (**Compressed View**). To return to an expanded view, click (**Expanded View**).
7. To limit the mode of the view, there are two buttons in the task bar.
   - Click (**Details Mode**) to see all details for an attribute.
   - Click (**Exists Mode**) to limit the view to if an attribute exists compared to the base or not. This only applies to attributes that can have a boolean property. For example, a user account exists or does not exist, or a piece of hardware that does or does not exist.
8. To change the base instance that all the others are compared to, click its label at the top of its column.
9. To go to the summary screen for an instance, click its **Virtual Thumbnail** or icon.

Creating an Instance Comparison Report

Output a the data from a comparison report in TXT, CSV or PDF formats.

1. Create the comparison for the report.
2. Click (**Download**).
   - Click for a **TXT** file.
   - Click for a **CSV** file.
   - Click for a **PDF** file.

Refreshing Instances and Images

Refresh your instances to get the latest data the provider can access. This includes information such as the power state, container, and hardware devices attached to the instance.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the desired items to analyze.
3. Click the checkboxes for the items to refresh.
4. Click **Configuration**, and then (**Refresh Relationships and Power States**) on the **Instance Taskbar**.

Extracting Running Processes from Instances and Images

Infrastructure Automation can collect processes running on Windows instances. To do this, enter domain credentials for the zone where the instance is located. The instance must be running and must have an IP address in the VMDB, usually obtained from a SmartState Analysis.

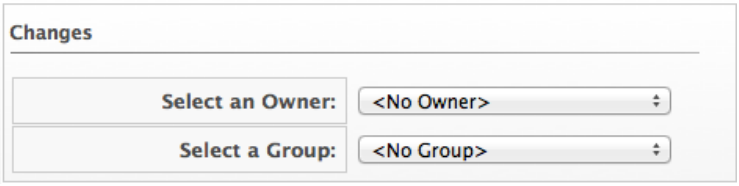
1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the checkboxes for the instances to collect processes.

3. Click **Configuration**, and then (**Extract Running Processes**) on the taskbar.
4. Click **OK**.

Setting Ownership for Instances and Images

You can set the owner of a group of instances and images by either individual user or group. This allows you an additional way to filter and can be used to enforce quotas.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the items to change.
3. Click the checkboxes for the items to set ownership.
4. Click **Configuration**, and then (**Set Ownership**) on the **Instance Taskbar**.
5. From the **Select an Owner** dropdown, select a user.



6. From the **Select a Group** dropdown, select a group
7. Click **Save**.

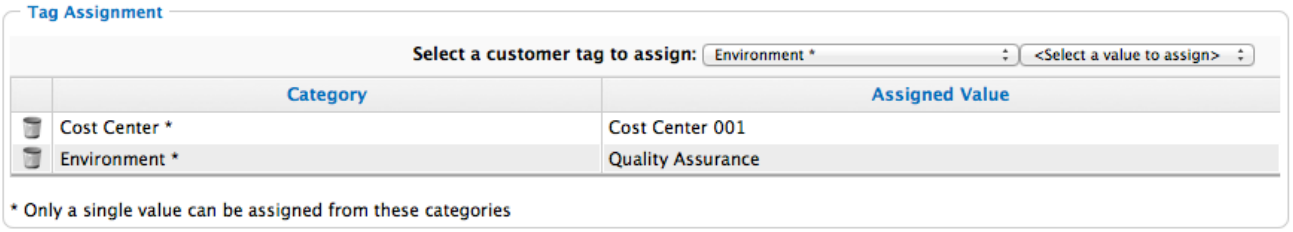
Removing Instances and Images from the VMDB

If an instance has been decommissioned or you need to perform some troubleshooting, you might need to remove a specific instance from the VMDB. This does not however remove the instance or image from its provider.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the items to remove.
3. Click the checkboxes for the items to remove.
4. Click **Configuration**, and then (**Remove from the VMDB**) button.
5. Click **OK**.

Tagging Instances and Images

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the items to tag.
3. Click the checkboxes for the items to tag.
4. Click (**Policy**), and then (**Edit Tags**).
5. Select a customer tag from the first dropdown, and then a value for the tag.



6. Click **Save**.

Reviewing an Instance or Image

After viewing your list of instances and images, click on a specific item to review a **Summary** screen of it. The **Summary** screen provides you with a **Virtual Thumbnail** and a **Taskbar**.

- Use the **Taskbar** to perform actions on the selected item.
- Use **Summary Views** to change the view type of the summary screen.
- Use **Virtual Thumbnails** for a quick glance at the item.
- Use the **Summary** screen to see a quick summary of the attributes of the item.

Viewing Running Processes after Collection

1. Click an instance with collected processes.
2. From the **Diagnostics** area, click **Running Processes**.

The most recent collection of running processes is displayed. Sort this list by clicking on the column headers.

Managing Security Groups for Cloud Instances

Manage security groups associated with cloud provider instances using Infrastructure Automation.

To add a security group to an instance:

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click on **Instances by Provider**, and select an instance.
3. Click **Configuration**, then click (**Add a Security Group**).
4. Select a **Security Group** from the drop-down menu to add to the instance.
5. Click **Save**.

To remove a security group from an instance:

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click on **Instances by Provider**, and select an instance.
3. Click **Configuration**, then click (**Remove a Security Group**).
4. Select a **Security Group** from the drop-down menu to remove from the instance.
5. Click **Save**.

Editing Instance or Image Properties

Edit the properties of an instance or image to set parent and child instances. SmartState Analysis also can detect this.

1. From **Compute > Clouds > Instances**.
2. Click the accordion for the items to edit.
3. Click the item to edit properties.
4. Click (**Configuration**), and then (**Edit this Instance or Edit this Image**) on the Taskbar.
5. From the **Parent Instance** dropdown, select the parent instance.
6. From **Child Instance** selection, select instances that are based on the current instance from the list of **Available Instances**.
7. Click **Save**.

Controlling the Power State of an Instance

Follow this procedure to control the power states of an instance through the Infrastructure Automation console.

1. Navigate to **Compute > Clouds > Instances**.
2. Click the instance to change the power state.
3. Click **Power Operations**, then click the button for the desired power operation.
   - Click (**Start**) to start the selected instances.
   - Click (**Terminate**) to terminate the selected instances.
   - Click (**Suspend**) to suspend the selected instances.
   - Click (**Reset**) to reset the selected instances.
   - Click (**Stop Guest**) to stop the guest operating system.
   - Click (**Restart Guest**) to restart the guest operating system.
4. Click **OK**.

Right Sizing an Instance

Infrastructure Automation uses collected statistics to recommend the best size for an instance. Infrastructure Automation uses the information from the **Normal Operating Range** to calculate the recommendations.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click an instance for right-sizing.
3. Click **Configuration**, and then (**Right-Size Recommendations**) button.

A new page appears with three levels of Memory and CPU recommendations, Conservative, Moderate, and Aggressive, next to the Normal Operating Range statistics.

Resizing an Instance

Infrastructure Automation allows you to resize an existing, active instance by changing its flavor. This is only possible if your OpenStack deployment has:

- At least two Compute nodes, or with resizing to the same host enabled
- Enough capacity to support the needs of the new flavor

Note:

Keep in mind that the instance will undergo a controlled shutdown when you change its flavor.

For more information about the requirements and underlying OpenStack process involved, see Resize an Instance in the Red Hat OpenStack Platform *Instances and Images Guide*.

To resize an instance through Infrastructure Automation:

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the instance whose flavor you want to change.
3. Click **Configuration**, and then (**Reconfigure this Instance**).
4. In the **Reconfigure Instance** section, select the new flavor you want from the **Choose Flavor** dropdown.
5. Click **Submit**. Doing so will initiate the flavor change, and it might take several minutes before Infrastructure Automation verifies whether the change was successful.

See Flavors and Manage Flavors in the Red Hat OpenStack Platform *Instances and Images Guide* for more information.

Migrating a Live Instance

Live migration involves moving a live instance between Compute nodes. Live migration is useful for avoiding instance downtime during cloud maintenance or load management. See How to Migrate a Live Instance in the Red Hat OpenStack Platform *Migrating Instances* guide for details on the underlying OpenStack process.

To migrate a live instance:

1. Browse to menu: **Compute > Clouds > Instances**.
2. On the right pane, click the instance to be migrated. Use the **Instances by Provider** accordion to filter instances by provider and/or availability zone.
3. Click (**Lifecycle**), then (**Migrate selected Instance**).
4. On the **Migrate Instance** section, select your preferred migration options:
   - **Auto-select Host?**: let the OpenStack provider automatically choose a destination Compute node. If you prefer to choose a specific node, uncheck this option and choose from the **Destination Host** dropdown.
   - **Block Migration**: check this option to perform a *block-based* migration. With this migration, the entire virtual machine image is moved from the source node to the destination node. If your OpenStack provider uses *shared storage*, leave this option unchecked. See Prerequisites in the Red Hat OpenStack Platform *Migrating Instances* guide for related information.
   - **Disk Over Commit**: check this option to prevent the OpenStack provider from verifying first whether the destination host has available disk space to host the instance.
5. Click **Submit**.

Once the migration initiates, the instance list will reload with a message indicating that the selected instance is being migrated. Upon completion, the instance list will reload and the evacuated instance will be displayed as (**On**).

Evacuating an Instance

If a Compute node is shut down, you can *evacuate* instances hosted on it. This is only useful if the instances use shared storage or block storage volumes. See Evacuate Instances in the Red Hat OpenStack Platform *Instances and Images Guide* for details on the underlying OpenStack process.

To evacuate an instance:

1. Browse to menu: **Compute > Clouds > Instances**.
2. On the right pane, click the instance to be evacuated. Use the **Instances by Provider** accordion to filter instances by provider and/or availability zone.
3. Click (**Lifecycle**), then (**Evacuate selected Instance**).
4. On the **Evacuate Host** section, select your preferred evacuation options:
   - **Auto-select Host?**: let the OpenStack provider automatically choose a destination Compute node. If you prefer to choose a specific node, uncheck this option and choose from the **Destination Host** dropdown.

- **On Shared Storage**: leave this checked to indicate that all instance files are on shared storage.

5. Click **Submit**.

Once the evacuation initiates, the instance list will reload with a message indicating that the selected instance is being evacuated. Upon completion, the instance list will reload and the evacuated instance will be displayed as (**On**).

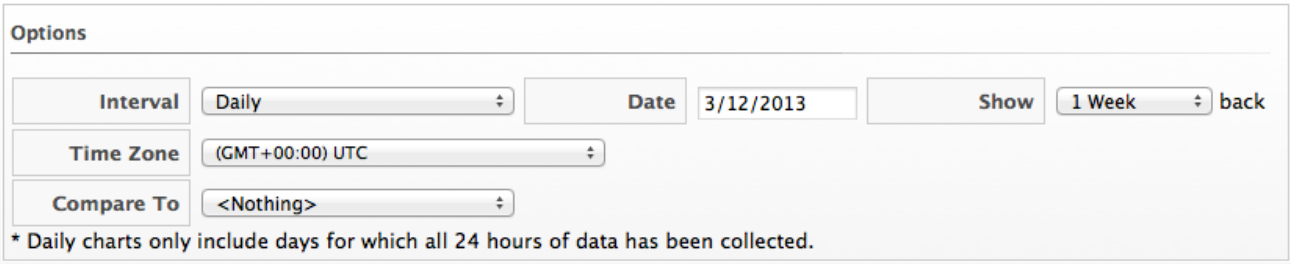
Viewing Capacity and Utilization Charts for an Instance

View capacity and utilization data for instances that are part of a cluster.

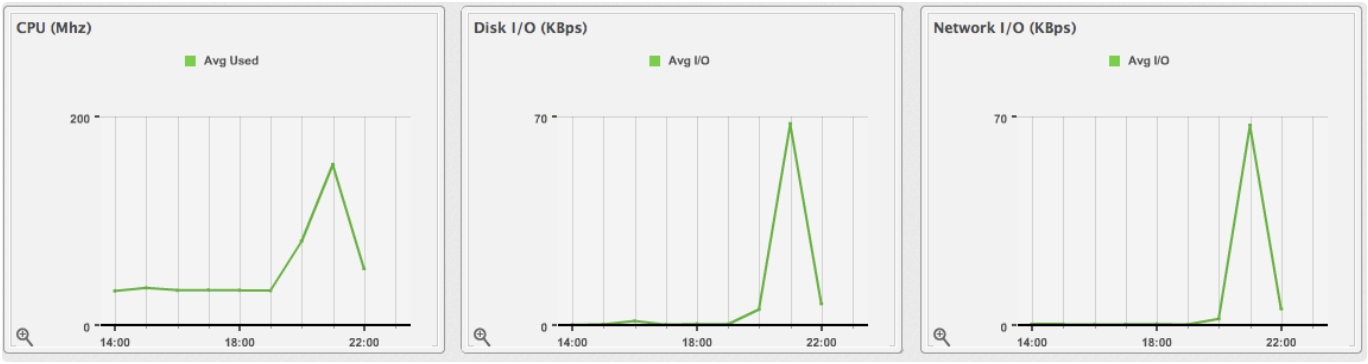
Note:

You must have a server with network visibility to your provider assigned the server role of **Capacity & Utilization Collector** to use this feature.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion to view capacity data.
3. Click the item to view.
4. Click (**Monitoring**), and then (**Utilization**) on the taskbar.
5. Select to view hourly, most recent hour, or daily data points for the dates to view data.



6. Select a **Time Profile**.



Note:

Daily charts only include full days of data. This means Infrastructure Automation does not show daily data for a day without a complete 24 data point range for a day.

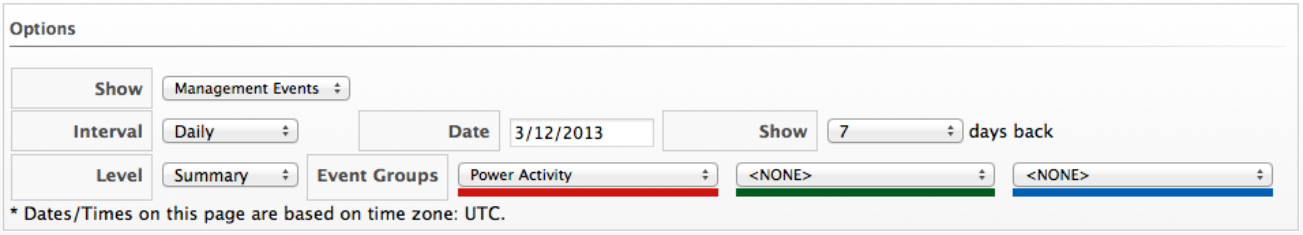
For information about data optimization including utilization trend reports, see Data Optimization.

Viewing the Instance or Image Timeline

View the timeline of events for an instance or image if registered to a host.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the instance to view the timeline.

3. Click (**Monitoring**), and then (**Timelines**) on the taskbar.
4. From **Options**, customize the period of time to display, and the types of events to view.



   - Use the **Interval** dropdown to select hourly or daily data points.
   - Use **Date** to type the date of the timeline to display.
   - If viewing a daily timeline, use **Show** to set how many days back to go. The maximum history is 31 days.
   - The three **Event Group** dropdowns allow selection of different event groups to display. Each has its own color.
   - From the **Level** dropdown, select either a **Summary** event or a **Detail** list of events. For example, the detail level of a **Power On** event might include the power on request, the starting event, and the actual **Power On** event. If you select **Summary**, you only see the **Power On** event in the timeline.
5. To see more detail on an item in the timeline, click on it. A balloon appears with a clickable link to the resource.

Viewing the Instance or Image Summary

When you click on a specific instance or image, you will see the **Virtual Thumbnail**, and an operating system-specific summary screen of the item. Where applicable, click on a subcategory of the summary to see more detail on that section.

The summary page contains the following categories:

- **Properties** include information such as the base operating system, hostname, IP addresses, instance vendor, cloud resources, and snapshots. This includes the ability to analyze multiple partitions, multiple disks, Linux logical volumes, extended partitions, and Windows drives. Some categories can be clicked on for additional detail. For example, click **Container** to view notes associated with an instance.
- **Lifecycle** shows the date of discovery and the last analysis. If a retirement date and time or owner has been set, these display as well.
- **Relationships** include information on the instance's cloud provider, genealogy such as parent and child instances, and drift.
- **VMsafe** shows properties of the VMsafe agent if it is enabled.
- **Compliance** shows the status of system compliance checks and history of past checks.
- **Power Management** displays the current power state, last boot time, and last power state change. **State Changed On** is the date that the instance last changed its power state. This is a container view of the power state, therefore a restart of the operating system does not cause the container power state to change and does not update this value.
- **Security** includes information on users and groups.
- **Configuration** includes information on applications, services, packages, init processes, and files. This section changes depending on the base operating system.
- **Diagnostics** provides a link to viewing running processes and the information from the latest collected event logs.
- **Smart Management** shows all tags assigned to this instance.

Performing a SmartState Analysis on an instance or image provides more detailed information in these categories.

Viewing User Information for an Instance or Image

Infrastructure Automation's **SmartState Analysis** feature returns user information. Explore the user to get details on the user's account, including group memberships.

1. Browse to menu: **Compute > Clouds > Instances**.

2. Click on the instance or image to open its summary page.
3. From the **Security** section of the summary page, click **Users**.
4. Click the user to view details.

Viewing Group Information for an Instance or Image

Infrastructure Automation's **SmartState Analysis** feature returns group information. Explore the group to get a list of its users.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the item to view user information.
3. Click on the item to view its **Summary**.
4. From the **Security** section of the **Instance Summary**, click **Groups**.
5. Click the group to view users.

Viewing Genealogy of an Instance or Image

Infrastructure Automation detects the lineage of an instance. View an instance's lineage and compare the instances that are part of its tree. This also allows tagging of instances that share genealogy.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the item to view genealogy.
3. Click on the item to view its **Summary**.
4. From the **Relationships** area in the **Summary**, click **Genealogy**.

Detecting Drift on Instances or Images

The configuration of an instance might change over time. **Drift** is the comparison of an instance to itself at different points in time. The instance needs to be analyzed at least twice to collect this information. Detecting drift provides you the following benefits:

- See the difference between the last known state of a machine and its current state.
- Review the configuration changes that happen to a particular instance between multiple points in time.
- Review the association changes that happen to a particular instance between multiple points in time.
- Review the classification changes that happen to an instance between two time checks.
- Capture the configuration drifts for a single instance across a time period.

Detect drift on instances or images:

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the item to view drift.
3. Click on the item to view its **Summary**.
4. From the **Relationships** area in the **Summary**, click **Drift History**.
5. Click the checkboxes for the analyses to compare.
6. Click (**Select up to 10 timestamps for Drift Analysis**)] at the top of the screen. The results display.
7. Check the **Drift** sections on the left to view in your comparison.
8. Click **Apply**.

9. The following descriptions pertain to the **Expanded View** . Whether you see the value of a property or an icon representing the property depends on the properties type.
    - A property displayed in the same color as the base means the compared analysis matches the base for that property.
    - A property displayed in a different color from the base means the compared analysis does not match the base for that property.
10. If you are in the **Compressed View** , the values of the properties are not displayed. All items are described by the icons shown below.
    - A (**checkmark**) means that the compared analysis matches the base for that property. If you hover over it, the value of the property will display.
    - A (**triangle**) means the compared analysis does not match the base for that property. If you hover over it, the value of the property displays. Click the minus sign next to the sections name to collapse it.
11. To limit the scope of the view, you have three buttons in the **Resource** button area.
    - Click (**All attributes**) to see all attributes of the sections you selected.
    - Click (**Attributes with different values**) to see only the attributes that are different across the drifts.
    - Click (**Attributes with the same values**) to see only the attributes that are the same across drifts.
12. To limit the mode of the view, there are two buttons in the **Resource** button area.
    - Click (**Details Mode**) to see all details for an attribute.
    - Click (**Exists Mode**) to only see if an attribute exists compared to the base or not. This only applies to attributes that can have a Boolean property. For example, a user account exists or does not exist, or a piece of hardware that does or does not exist.

This creates a drift analysis. Download the data or create a report from your drift for analysis using external tools.

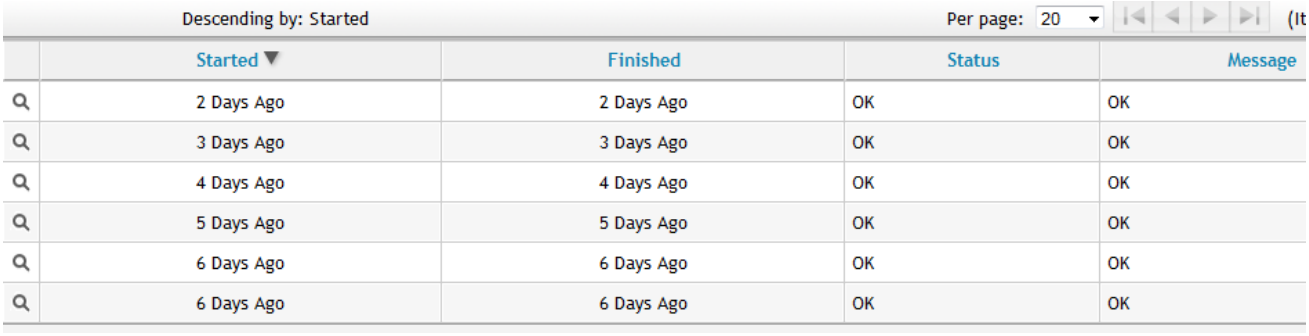
Creating a Drift Report for an Instance or Image

1. Create the comparison to analyze.
2. Click (**Download**).
3. Click the output button for the type of report you want.
    - Click (**Download drift report in text format**) for a text file.
    - Click (**Download drift report in CSV format**) for a csv file.
    - Click (**Download drift report in PDF format**) for a PDF file.

Viewing Analysis History for an Instance or Image

Each time a SmartState Analysis is performed on an instance, a record is created of the task. This information is accessed either from the instance accordion or the instance summary. Use this detail to find when the last analysis was completed and if it completed successfully. If the analysis resulted in an error, the error is shown here.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the desired item to view analysis history.
3. Click on the item to view its summary.
4. From the **Relationships** area in the summary, click **Analysis History**. A history of up to the last 10 analyses is displayed.



Descending by: Started — Per page: 20 — (It

	Started ▼	Finished	Status	Message
🔍	2 Days Ago	2 Days Ago	OK	OK
🔍	3 Days Ago	3 Days Ago	OK	OK
🔍	4 Days Ago	4 Days Ago	OK	OK
🔍	5 Days Ago	5 Days Ago	OK	OK
🔍	6 Days Ago	6 Days Ago	OK	OK
🔍	6 Days Ago	6 Days Ago	OK	OK

5. Click on a specific analysis to see its details.

Viewing Event Logs for an Instance or Image

Using an **Analysis Profile**, collect event log information from your instances.

Note:

This feature is only available for Windows.

1. Browse to menu: **Compute > Clouds > Instances**.
2. Click the accordion for the item to view event logs.
3. Click on the item to view its **Summary**.
4. From **Diagnostics** click **Event Logs**.

The collected event log entries are displayed. Sort this list by clicking on the column headers.

Orchestration Stacks

Browse to menu: **Compute > Clouds > Stacks** to see a list of orchestration stacks along with information such as *Name*, *Provider*, *Type*, *Status*, *Instances*, *Security Groups*, and *Cloud Networks*. Click on a stack to see more information about it, including properties, retirement date and time, and relationships with the cloud provider and instances. You can click on instances to see details of all instances the stack relates to.

Tagging Orchestration Stacks

Apply tags to orchestration stacks to categorize them together at the same time. . Browse to menu: **Compute > Clouds > Stacks**. . Select the orchestration stacks to tag. . Click (**Policy**), and then (**Edit Tags**). . **Select a customer tag to assign** from the first list. . Select a value to assign from the second list. . Click **Save**.

Retiring Orchestration Stacks

You can either retire orchestration stacks on a set date and time, or retire them immediately.

To set a retirement date:

1. Browse to menu: **Compute > Clouds > Stacks**.
2. Select the orchestration stacks to retire on a set date and time.
3. Click (**Lifecycle**), and then (**Set Retirement Dates**).
4. On the **Retire Orchestration Stacks** page, set **Retirement Date and Time**.
5. Select **Retirement Warning** from the list. The default is *None*.

6. Click **Save**.

Note:

Saving a blank date will remove all retirement dates.

To retire selected stacks immediately:

1. Browse to menu: **Compute > Clouds > Stacks**.
2. Select the orchestration stacks to retire now.
3. Click (**Lifecycle**), and then (**Retire selected Orchestration Stacks**). A pop-up window appears to confirm the action.
4. Click **OK**.

Removing Orchestration Stacks

1. Browse to menu: **Compute > Clouds > Stacks**.
2. Select the orchestration stacks to remove from the VMDB. A warning pop-up window appears to confirm the action.
3. Click **Configuration**, and then (**Remove Stacks from the VMDB**).
4. Click **OK**.

Key Pairs

This tab lists key pairs and fingerprints for all cloud providers. Click on a key pair to see a summary and its relationship with instances. On this screen, click on instances to see details of all instances the key pair relates to. You can use the key pairs added during provisioning instances.

Note:

Adding a new key pair is currently only supported for OpenStack.

Adding a New Key Pair

1. Browse to menu: **Compute > Clouds > Key Pairs**.
2. Click **Configuration**, then click (**Add a new Key Pair**).
3. In **Basic Information**, enter a **Name** and the **Public Key (optional)** generated using **ssh-keygen** command.
4. Select your OpenStack provider from the **Provider** list.
5. Click **Add**.

Removing a Key Pair

1. Browse to menu: **Compute > Clouds > Key Pairs**.
2. Select the key pair you want to remove from the key pairs list. Or, click on the key pair to see the instances it relates to.
3. Click **Configuration**, then click (**Remove selected Key Pairs from Inventory**). A warning appears to confirm the action.
4. Click **OK**.

Downloading Key Pairs

1. Browse to menu: **Compute > Clouds > Key Pairs**. You will see a list of existing key pairs.
2. Click the button, then select the option to download key pairs data in your preferred format:
    1. Download as Text
    2. Download as CSV
    3. Print or export as PDF

Object Stores

Browse to menu: **Storage > Object Stores** to see a list of all cloud object stores along with information including *Key*, *Size (bytes)*, *Object Count*, *Cloud Tenant*, and *Cloud Provider*. Click on an object store to see its properties and relationships with the cloud provider, tenant, and object store on the summary page.

Tagging Object Stores

Apply tags to object stores to categorize them together at the same time.

1. Browse to menu: **Storage > Object Stores**.
2. Select the object stores to tag.
3. Click **Policy**, and then **Edit Tags**.
4. **Select a customer tag to assign** from the first list.
5. Select a value to assign from the second list.
6. Click **Save**.

VMware Networking Switches

After adding a VMware provider, Infrastructure Automation automatically discovers all vSphere distributed switches (vDS) on that provider and collects the information in the Infrastructure Automation inventory.

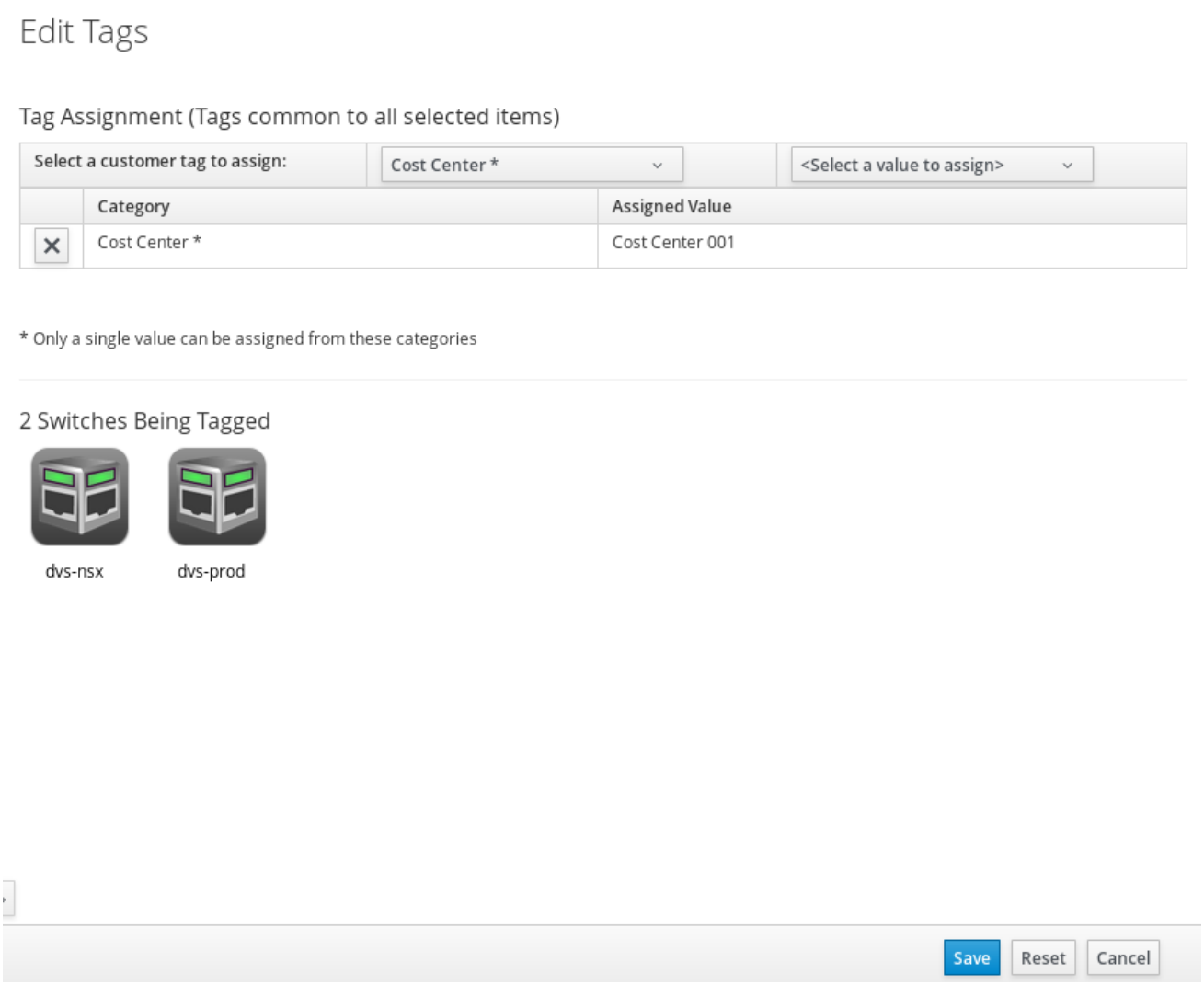
Browse to menu: **Compute > Infrastructure > Networking** to see a list of all VMware switches, along with information including *Name*, *Ports*, and *UUID*. Switches and port groups are listed by provider, then cluster on the sidebar.

Click on a switch to view its summary page, which displays relationships with hosts, and any tags.

Tagging VMware Networking Switches

Use tags to categorize vSphere distributed switches.

1. Browse to menu: **Compute > Infrastructure > Networking**.
2. Select the switches to tag.
3. Click (**Policy**), and then (**Edit Tags**).



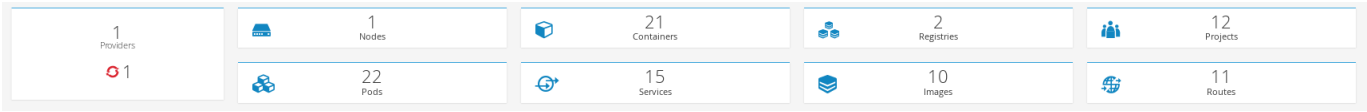
4. Select a customer tag from the first dropdown, and then a value for the tag.
5. Select more tags or click **Save** to apply the tags.

Container Entities

This chapter provides information on managing resources on your containers providers.

The Containers Overview Page

The containers overview page shows information on all containers providers and entities known to Infrastructure Automation. The **Overview** page provides links to other summary pages which contain further information on the containers providers and entities. The **Overview** page also provides metrics for **Aggregated Node Utilization**, **Network Utilization Trend**, **New Image Usage Trend**, **Node Utilization**, and **Pod Creation and Deletion Trends**.



1. Browse to menu: **Compute > Containers > Overview**.
2. Click the desired containers entity, or provider, if applicable, for viewing the summary with further information.

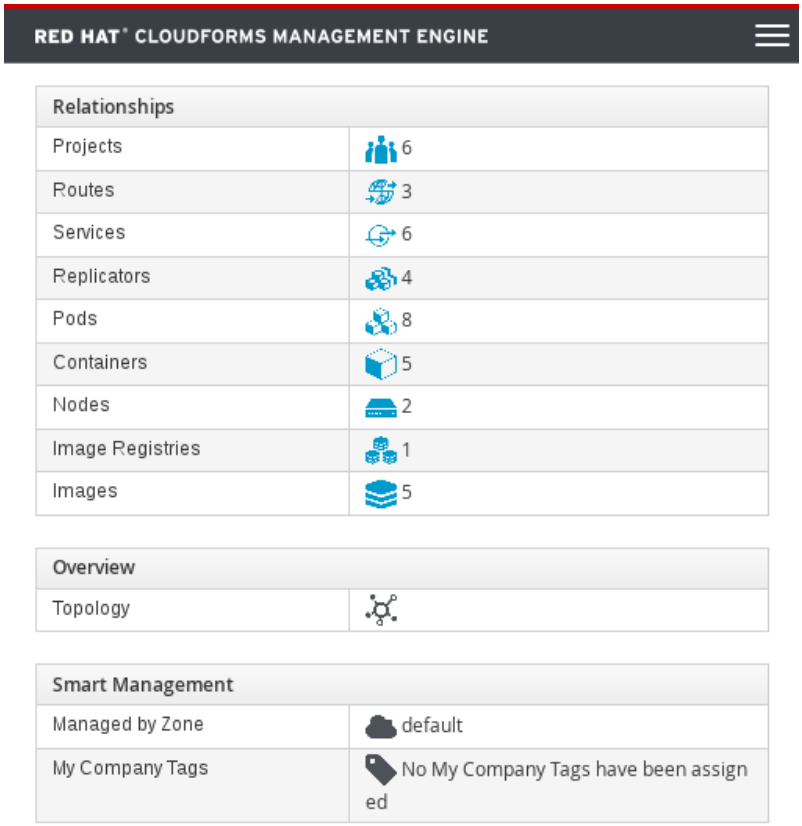
Note:

To reliably associate pods and images, Infrastructure Automation requires information from the **`docker-pullable`** field, added in OpenShift 3.3.1.2. This can affect the results of the **Chargeback by Image** report for older OpenShift providers, and potentially cause image inspection (done as part of Smart State Analysis) to fail due to associating a container to the wrong image. Consequently, Infrastructure Automation may not report accurate information about pods and images in OpenShift providers before version 3.3.1.2.

Viewing a Container Entity Summary

Container entity (object) summaries are found at menu: **Compute > Containers > Entity**, where you can view information about container entities and their components.

- Viewing a Containers Provider Summary Browse to menu: **Compute > Containers > Providers** to view information on different aspects of a containers provider. The summary includes:
    - The status of the provider and its components.
    - The relationships between different entities of the containers provider. These relationships are summarized in the **Relationships** box on the right-hand side of the summary page.



    - Additional information on aggregated capacity of all CPU cores of all nodes, and aggregated capacity of all memory of all nodes.
- Viewing a Container Nodes Summary Browse to menu: **Compute > Containers > Container Nodes** to view information on different aspects of a container node. The summary includes:
    - The number of entities on a node.
    - A node's capacity and utilization.
    - The version of the underlying operating system and software.

To view the timeline of events for a node from a container nodes summary page, click (**Monitoring**), and then (**Timelines**).

- Viewing a Containers Summary Browse to menu: **Compute > Containers > Containers** to view information on different aspects of a container. The summary includes:
    - The relationships of the container to a related node, pod, or image.
    - The node the container runs on.
    - The container ID.

    - Properties of the container image, such as name, tag, etc.
- Viewing a Container Images Summary Browse to menu: **Compute > Containers > Container Images** to view information on different aspects of a container image. The summary includes:
    - The containers currently using the images.
    - The image registry the image is from.
- Viewing an Image Registries Summary Browse to menu: **Compute > Containers > Image Registries** to view information on different aspects of an image registry. The summary includes:
    - Which images are from the registry.
    - The number of images that come from that registry.
    - Which containers use images from that registry.
    - The host and port of the registry.
- Viewing a Pods Summary Browse to menu: **Compute > Containers > Pods** to view information on different aspects of a pod. The summary includes:
    - The containers that are part of the pod.
    - The services that reference the pod.
    - The node the pod runs on.
    - If the pod controlled by a replicator.
    - The IP address of the pod.
- Viewing a Replicators Summary Browse to menu: **Compute > Containers > Replicators** to view information on different aspects of a replicator. The summary includes:
    - The number of requested pods.
    - The number of current pods.
    - The labels and selector for the replicator.
- Viewing a Container Services Summary Browse to menu: **Compute > Containers > Container Services** to view information on different aspects of a container service. The summary includes:
    - The pods that the container service provide traffic to.
    - The port configurations for the container service.
    - The labels and selector for the container service.
- Viewing a Volumes Summary Browse to menu: **Compute > Containers > Volumes** to view information on the persistent volumes of a container provider. The summary includes:
    - The pods the volume is connected to.
    - The volume's connection parameters.
    - The volume's storage capacity.
    - The volume's iSCSI target details (if applicable).
- Viewing a Container Builds Summary Browse to menu: **Compute > Containers > Container Builds** to view different aspects of a container build. The summary includes:
    - The build configuration the container build is based on.
    - Which build instances have been created.
    - Which phase in the build process the instance has completed.
    - Which pod a build instance reside in.
- Viewing a Container Templates Summary Browse to menu: **Compute > Containers > Container Templates** to view different aspects of a container template. The summary includes:
    - The project the template is associated with.

- The objects the template contains.
- The parameters that can be used with the template's objects.
- The template's version number.

Using the Topology Widget

The **Topology** widget is an interactive topology graph, showing the status and relationships between the different entities of the containers providers and projects to which Infrastructure Automation has access.

- The topology graph includes pods, containers, services, nodes, virtual machines, hosts, routes, and replicators within the overall containers provider environment. - Each entity in the graph displays a color indication of its status.
- Hovering over any individual graph element will display a summary of details for the individual element.
- Double-click the entities in the graph to browse to their summary pages.
- It is possible to drag elements to reposition the graph.
- Click the legend at the top of the graph to show or hide entities.
- Click **Display Names** on the right-hand side of the page to show or hide entity names.

Viewing the Topology for Container Providers

1. Browse to menu: **Compute > Containers > Providers**.
2. Click the desired containers provider for viewing the provider summary.
3. On the provider summary page, click **Topology** in the **Overview** box on the right side of the page.

Viewing the Topology for Container Provider Projects

The project topology page displays the project as the center node, surrounded by its related entities.

1. Browse to menu: **Compute > Containers > Projects**.
2. Click on a project.
3. On the project summary page, click (**Topology View**) on the top right side of the page.

Limiting the Number of Containers Shown in the Topology View

1. Navigate to the settings menu, then **My Settings**, and click on the **Visual** tab.
2. Select the number of container items from the drop-down under **Topology Default Items in View**.
3. Click **Save**.

Analyzing Container Images with SmartState Analysis

Perform a SmartState Analysis of a container image to inspect the packages included in an image.

1. Browse to menu: **Compute > Containers > Container Images**.
2. Check the container image to analyze. You can check multiple images.
3. Click **Configuration**, and then **Perform SmartState Analysis**.

The container image is scanned. The process will copy over any required files for the image. After reloading the image page, all new or updated packages are listed.

To monitor the status of container image SmartState Analysis tasks, navigate to the settings menu, then **Tasks**. The status of each task is displayed including time started, time ended, what part of the task is currently running, and any errors encountered.

Note:

See *Scanning Container Images in with OpenSCAP* for details on scanning container images using OpenSCAP policies.

Configuring Automatic Tagging for Container Entities

Container object labels in OpenShift can be used to automatically create tags and tag categories in Infrastructure Automation. This is done by mapping Infrastructure Automation tags to existing OpenShift or Kubernetes labels.

Labels from OpenShift can be mapped to Infrastructure Automation tags for the following container entities:

- Projects
- Nodes
- Routes
- Replicators
- Container services
- Pods
- Container builds

Note:

Tags automatically created from OpenShift labels are completely managed by the Infrastructure Automation system and cannot be manually assigned or unassigned. Deleting a mapping rule from Infrastructure Automation immediately deletes the resulting tags.

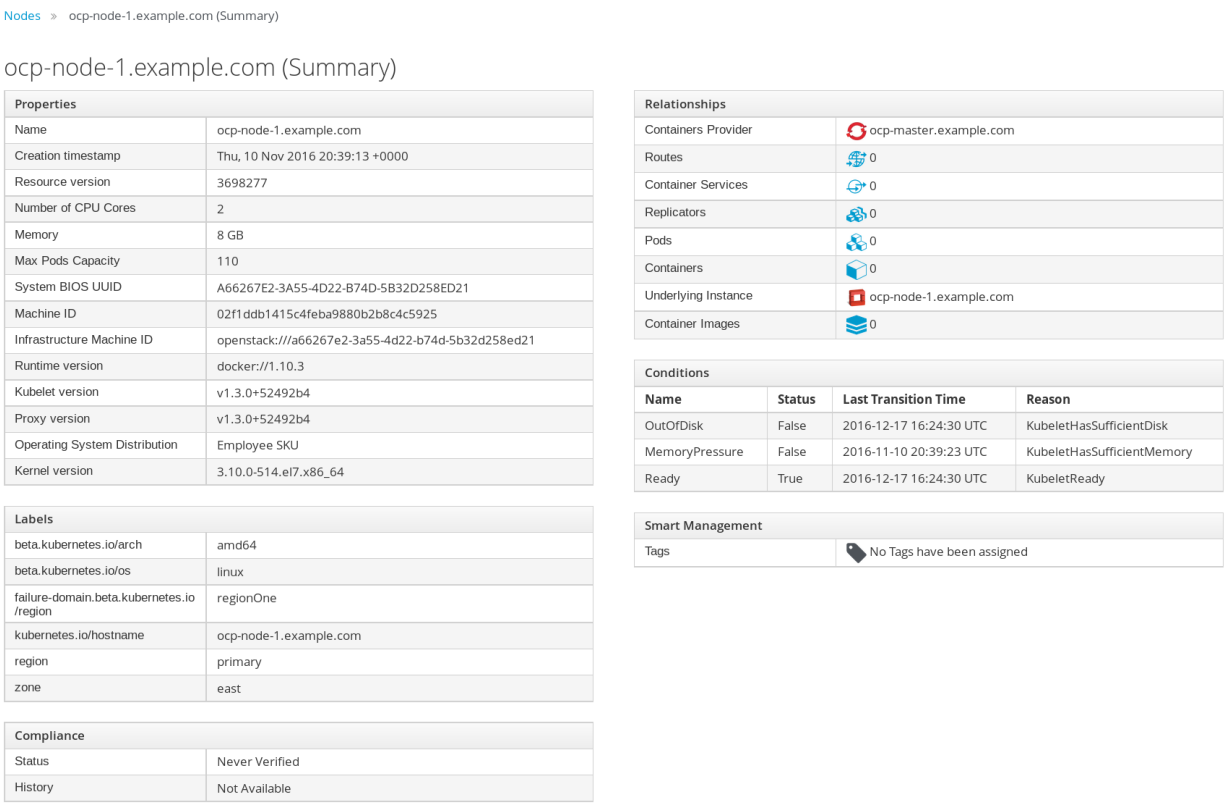
You can view a container entity's OpenShift labels on the entity's details page under **Labels**.

The following example shows how to configure tagging for a node, but the same steps can be used for mapping labels to tags on other container entities.

To configure automatic tagging on container entities using labels:

1. Note the *key* of the OpenShift label you want to map to a Infrastructure Automation tag. OpenShift labels consist of two parts: a *key* and a *value*.
    1. Browse to menu: **Compute > Containers > Container Nodes**.
    2. Select a node to open its summary page.
    3. Under **Labels**, note the label(s) to map to Infrastructure Automation tag(s). Any OpenShift labels will list the *key* in the left column of the **Labels** table, and the *value* in the right column of the **Labels** table.

       This node has six labels (key/value pairs) that were created in OpenShift and collected in the Infrastructure Automation inventory:


Nodes » ocp-node-1.example.com (Summary)

ocp-node-1.example.com (Summary)

Properties	
Name	ocp-node-1.example.com
Creation timestamp	Thu, 10 Nov 2016 20:39:13 +0000
Resource version	3698277
Number of CPU Cores	2
Memory	8 GB
Max Pods Capacity	110
System BIOS UUID	A66267E2-3A55-4D22-B74D-5B32D258ED21
Machine ID	02f1ddb1415c4feba9880b2b8c4c5925
Infrastructure Machine ID	openstack:///a66267e2-3a55-4d22-b74d-5b32d258ed21
Runtime version	docker://1.10.3
Kubelet version	v1.3.0+52492b4
Proxy version	v1.3.0+52492b4
Operating System Distribution	Employee SKU
Kernel version	3.10.0-514.el7.x86_64

Labels	
beta.kubernetes.io/arch	amd64
beta.kubernetes.io/os	linux
failure-domain.beta.kubernetes.io /region	regionOne
kubernetes.io/hostname	ocp-node-1.example.com
region	primary
zone	east

Compliance	
Status	Never Verified
History	Not Available

Relationships	
Containers Provider	ocp-master.example.com
Routes	0
Container Services	0
Replicators	0
Pods	0
Containers	0
Underlying Instance	ocp-node-1.example.com
Container Images	0

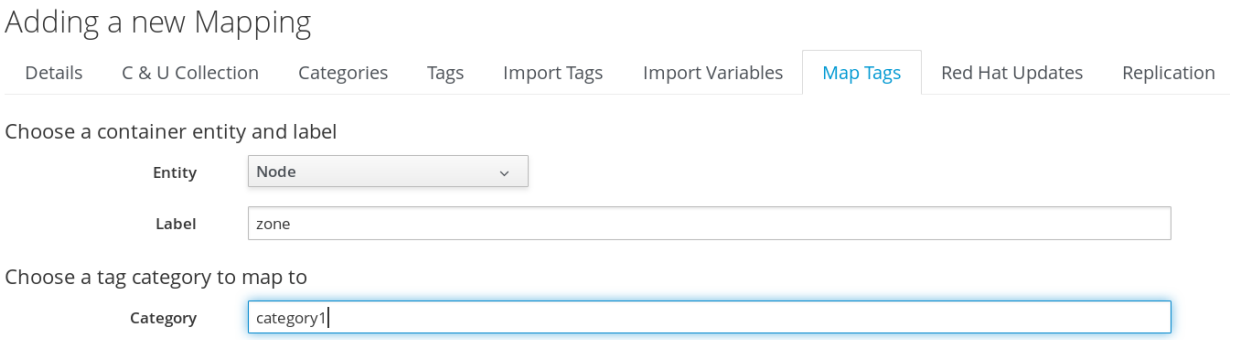
Conditions			
Name	**Status**	**Last Transition Time**	**Reason**
OutOfDisk	False	2016-12-17 16:24:30 UTC	KubeletHasSufficientDisk
MemoryPressure	False	2016-11-10 20:39:23 UTC	KubeletHasSufficientMemory
Ready	True	2016-12-17 16:24:30 UTC	KubeletReady

Smart Management	
Tags	No Tags have been assigned

Note:

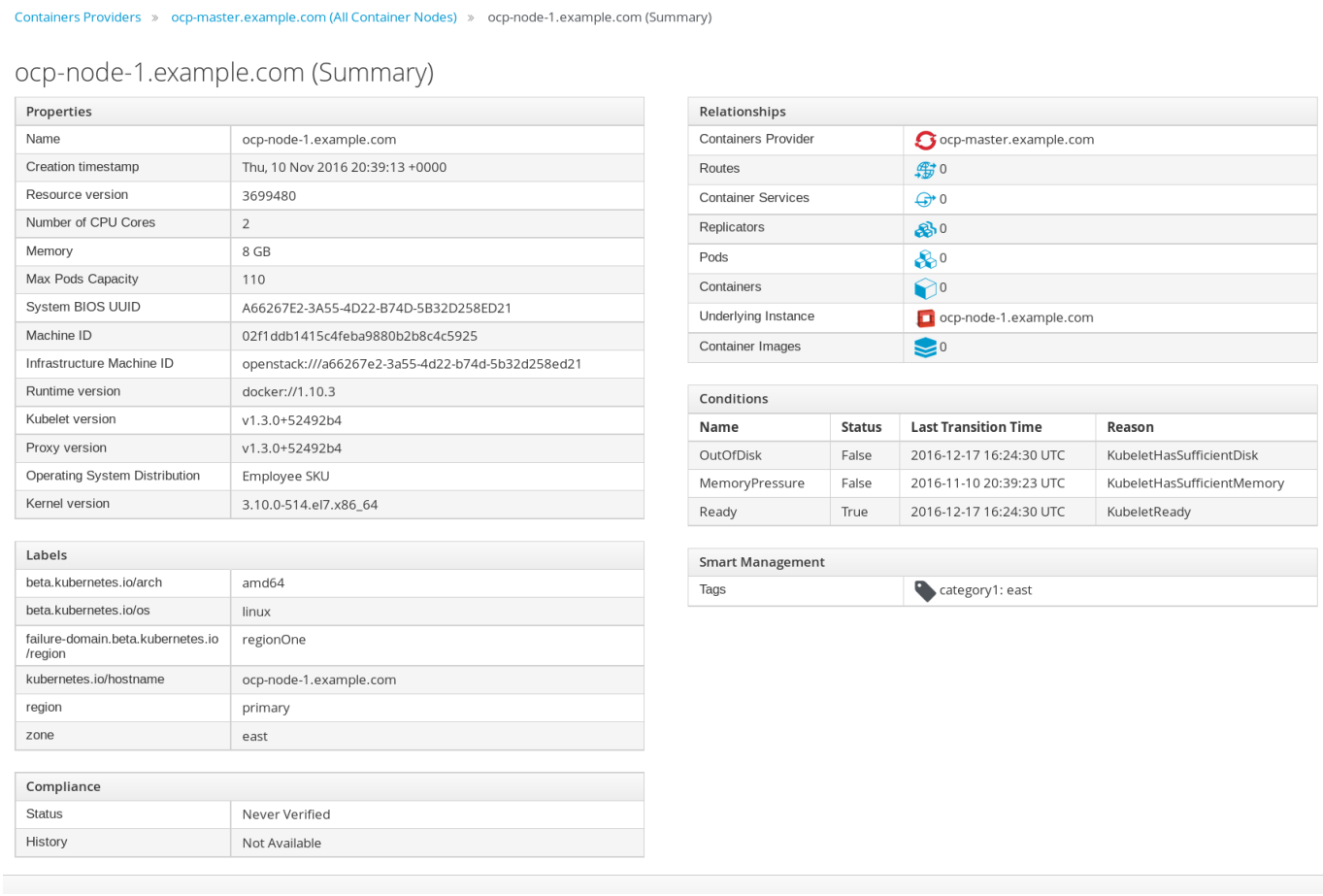
To create an OpenShift label, see Developer CLI Operations in the OpenShift Container Platform *CLI Reference* guide. A new label added in OpenShift will only show up in Infrastructure Automation after the next OpenShift provider refresh.

2. Navigate to **Configuration** and select the region.
3. Click the **Map Tags** tab.
4. Click **Add** to create a new mapping rule.
    1. Select a container entity to tag from the **Entity** list, or select **All** to tag all entities.
    2. Specify the *key* from the OpenShift label you noted earlier in the **Label** field.
    3. Specify a Infrastructure Automation tag category in **Category** to map the label to. If the tag category does not exist yet in Infrastructure Automation, it will be created automatically.



    4. Click **Add**. The mapping will show in the table on the **Map Tags** tab.
5. Refresh the provider to complete the mapping:
    1. Browse to menu: **Compute > Containers > Providers**.
    2. Select the provider to refresh.
    3. Click **Configuration**, and then **Refresh Items and Relationships**.

The label will display on the entity's summary page under **Smart Management** under **Company Tags** as `<Category> : <value>`.



Any container entity with the OpenShift **`zone`** label will be tagged automatically as **`category1`** in Infrastructure Automation. If the *value* for **`zone`** is **`south`**, for example, the entity will be tagged as `category1
south`.

You can use these tags to create reports. See Monitoring, Alerts, and Reporting for details on creating reports.

Managing controls

Learn more about managing controls for Infrastructure Automation.

- Policies and Profiles Guide
- Administration
- Monitoring, Alerts, and Reporting

- Policies and Profiles Guide
  - Policies
    - Control Policies
      - Creating Control Policies
      - Editing Basic Information, Scope, and Notes for a Policy
      - Copying a Policy
      - Deleting a Policy
      - Creating a New Policy Condition
      - Editing Policy Condition Assignments
      - Editing Policy Event Assignments
      - Assigning an Action to an Event
    - Compliance Policies
      - Creating a Compliance Policy
      - Creating a Compliance Condition to Check Host File Contents
      - Checking for Compliance
        - Scheduling a Compliance Check
        - Checking a Virtual Machine for Compliance from the Summary Screen

        - Checking a Host for Compliance from the Summary Screen
        - Checking a Replicator for Compliance from the Summary Screen
        - Checking a Pod for Compliance from the Summary Screen
        - Checking a Container Node for Compliance from the Summary Screen
        - Checking a Container Image for Compliance from the Summary Screen
    - Conditions
        - Creating a Condition
        - Editing a Condition
        - Copying a Condition
        - Deleting a Condition
    - Actions
        - Custom Actions
            - Creating an Assign Profile to Analysis Task Action
            - Creating a Snapshot Action
            - Deleting Snapshots by Age
            - Evaluating an Alert
            - Creating an Inherit Tag Action
            - Creating a CPU Reconfigure Action
            - Creating a Memory Reconfigure Action
            - Creating a Remove Tag Action
            - Creating an Ansible Playbook Run Action
            - Creating an E-mail Action
            - Creating an SNMP Action
            - Creating a Set Custom Attribute Action
            - Creating a Tag Action
        - Editing an Action
        - Deleting an Action
    - Policy Profiles
        - Creating Policy Profiles
        - Deleting a Policy Profile
        - Simulating Policy
            - Simulating Policy Profiles on Virtual Machines
        - Assigning Policy Profiles
            - Assigning Policy Profiles to an Infrastructure Provider
            - Removing Policy Profiles from an Infrastructure Provider
            - Assigning Policy Profiles to a Cluster
            - Removing Policy Profiles from a Cluster
            - Assigning Policy Profiles to a Host
            - Removing Policy Profiles from a Host
            - Assigning Policy Profiles to a Virtual Machine
            - Removing Policy Profiles from a Virtual Machine
            - Assigning Policy Profiles to a Resource Pool
            - Removing Policy Profiles from a Resource Pool
            - Assigning Policy Profiles to a Cloud Provider
            - Removing Policy Profiles from a Cloud Provider
            - Assigning Policy Profiles to a Network Provider
            - Removing Policy Profiles from a Network Provider
            - Assigning Policy Profiles to a Container Provider
            - Removing Policy Profiles from a Container Provider
            - Assigning Policy Profiles to a Replicator
            - Removing Policy Profiles from a Replicator
            - Assigning Policy Profiles to a Pod
            - Removing Policy Profiles from a Pod
            - Assigning Policy Profiles to a Container Node
            - Removing Policy Profiles from a Container Node
            - Assigning Policy Profiles to a Container Image
            - Removing Policy Profiles from a Container Image
            - Assigning Policy Profiles to an Instance
            - Removing Policy Profiles from an Instance
        - Disabling a Policy in a Policy Profile
        - Viewing Policy Simulation - Resultant Set of Policy (RSOP)
        - Exporting and Importing Analysis Profiles

Policies and Profiles Guide

Policies

Policies are used to manage your virtual environment. There are two types of policies available: compliance and control. Compliance policies are used to harden your virtual infrastructure, making sure that your security requirements are adhered to. Control policies are used to check for a specific condition and perform an action based on the outcome. For example:

- Prevent virtual machines from running without an administrator account.
- Prevent virtual machines from starting if certain patches are not applied.
- Configure the behavior of a production virtual machine to only start if it is running on a production host.
- Force a SmartState Analysis when a host is added or removed from a cluster.

Control Policies

A control policy is a combination of an event, a condition, and an action. This combination provides management capabilities in your virtual environment.

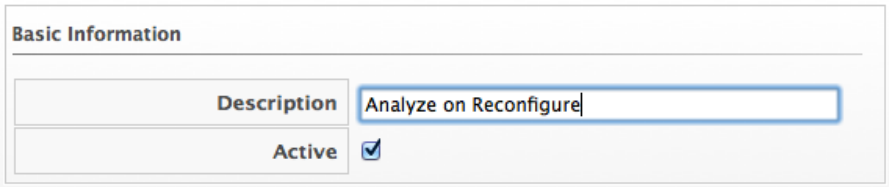
- An event is a trigger to check a condition.
- A condition is a test that is triggered by an event.
- An action is an execution that occurs if a condition is met.

Creating Control Policies

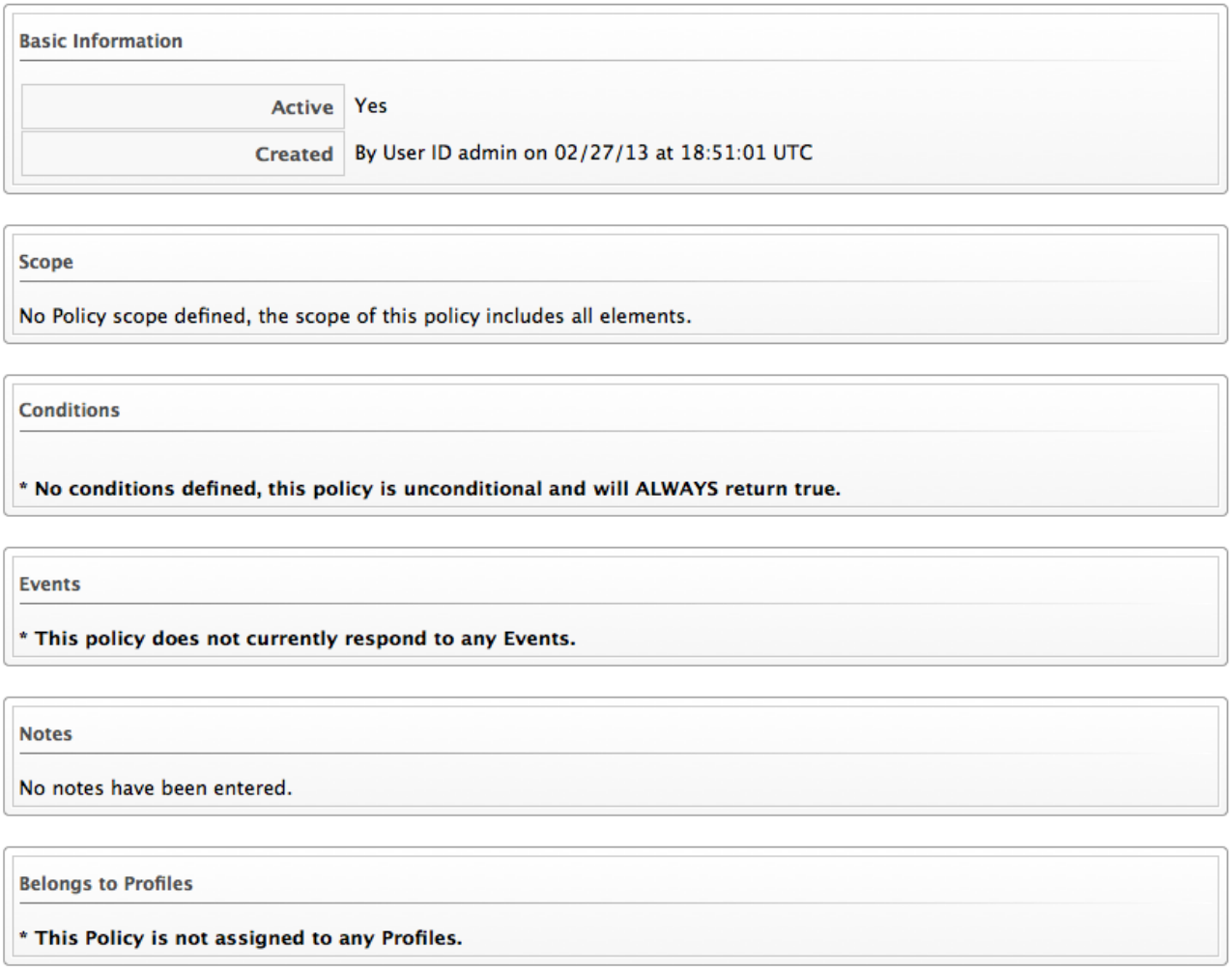
Create control policies by combining an event, a condition, and an action. Plan carefully the purpose of your policy before you create the policy. You can also use a scope expression that is tested immediately when the policy is triggered by an event. If the item is out of scope, then the policy does not continue on to the conditions, and none of the associated actions run.

This procedure describes how to create a control policy, its underlying conditions, and assign its events and actions in one process. Conditions and custom actions can be created separately as well. Those procedures are described in later sections in conditions and actions. Also, a description of all events is provided in events.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, and select **Control Policies**.
3. Select either **Host Control Policies** or **VM Control Policies** or **Replicator Control Policies** or **Pod Control Policies** or **Container Node Control Policies** or **Container Image Control Policies**.
4. Click **Configuration**, **Add a New Host / VM / Replicator / Pod / Node / Image Control Policy**.
5. Type in a **Description**.



6. Uncheck **Active** if you do not want this policy to be processed even when assigned to a resource.
7. You can enter a **Scope** here (You can also create a scope as part of a condition, or not use one at all). If the host or virtual machine is not included in the scope, no actions are run.
8. In the **Notes** area, add a detailed explanation of the policy.
9. Click **Add**. You are brought to the page where you add conditions and events to your new policy.

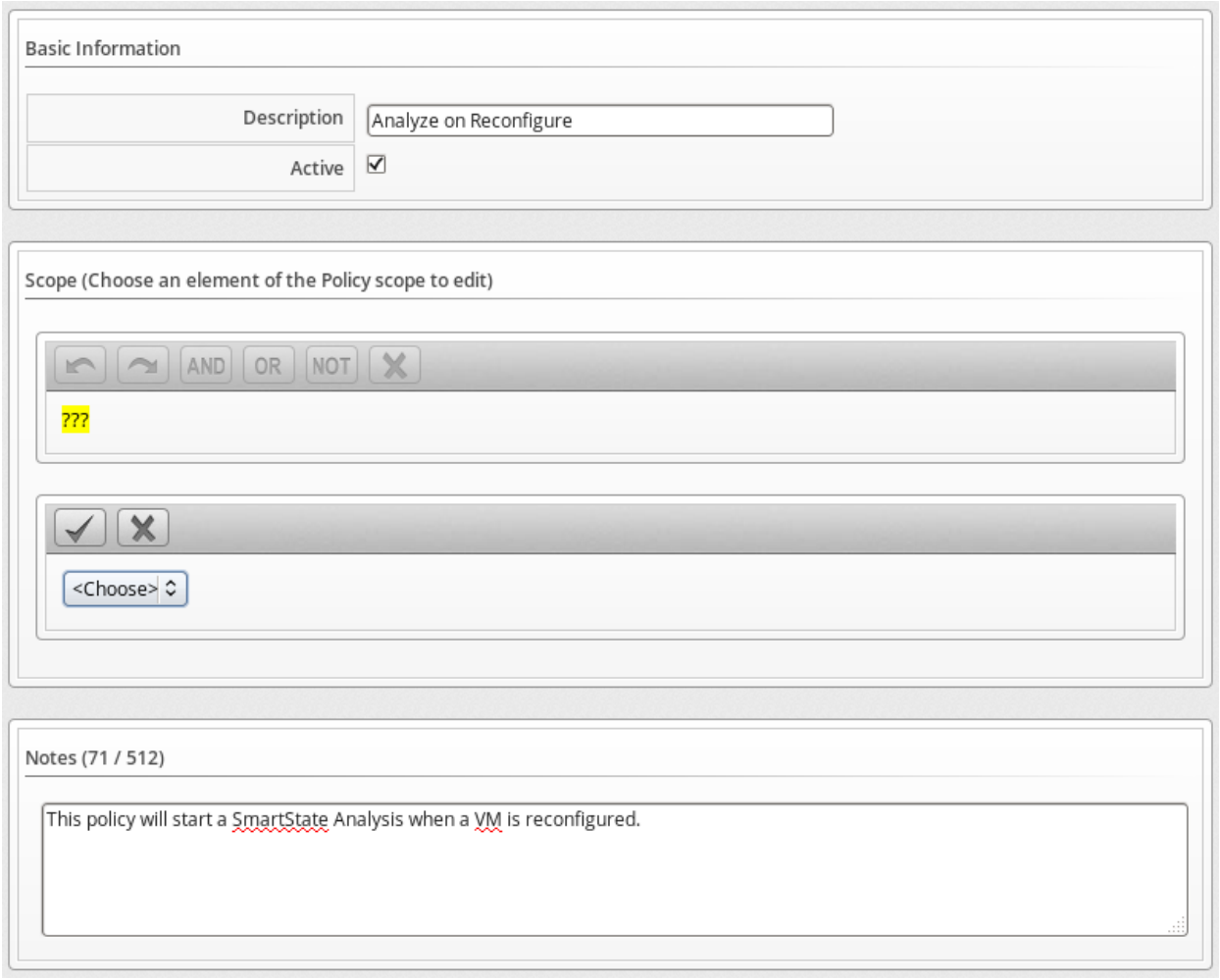


10. Click **Configuration** to associate conditions, events, and actions with the policy.

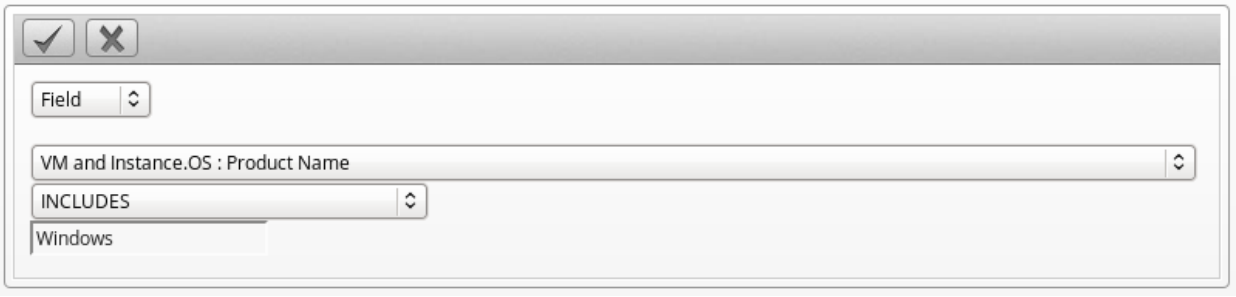
Editing Basic Information, Scope, and Notes for a Policy

As your enterprise's needs change, you can change the name of a policy or its scope. If the items that are being evaluated are out of scope, policy processing stops and no actions run.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, and select the policy to edit.
3. Click **Configuration**, **Edit Basic Info, Scope, and Notes**.
4. In the **Scope** area, create a general condition based on a simple attribute. Or, click an existing expression to edit it. Based on what you choose, different options appear. Configuring a **Scope** is optional for a policy.

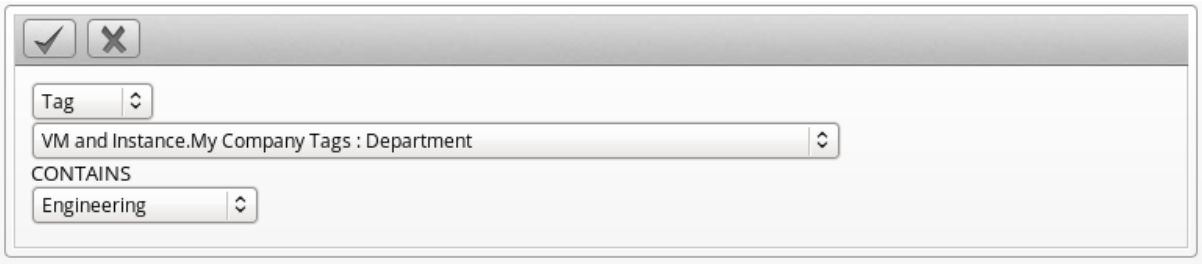


- Click **Field** to create criteria based on field values.

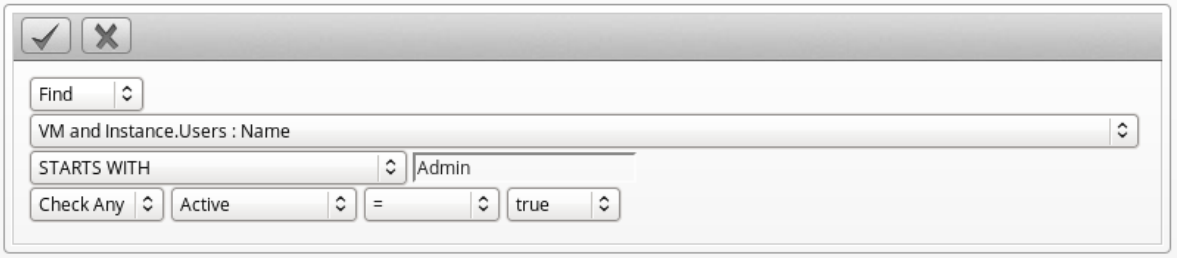


- Click **Count of** to create criteria based on the count of something, such as the number of snapshots for a virtual machine, or the number of virtual machines on a host.

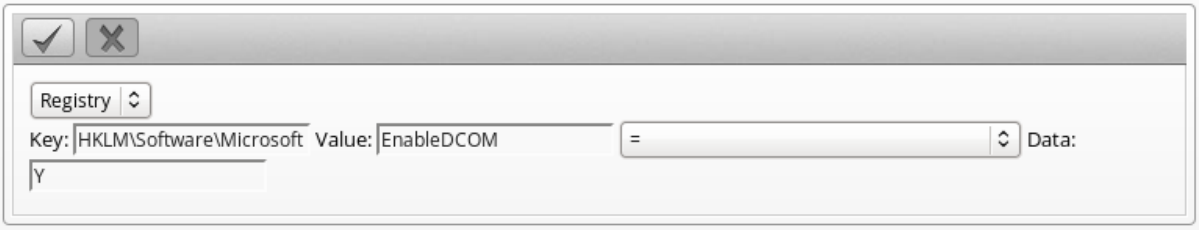
- Click **Tag** to create criteria based on tags that are assigned to your resources. For example, you can check the power state of a virtual machine or see whether it is tagged as production.



- Click **Find** to seek a particular value, and then check a property. For example, finding the **Admin** account and checking that it is enabled. Use the following check commands:
  - **Check Any**: The result is true if one or more of the find results satisfy the check condition.
  - **Check All**: All of the find results must match for a true result.
  - **Check Count**: If the result satisfies the expression in check count, the result is true.



- Click **Registry** to create criteria based on registry values. For example, you can check whether DCOM is enabled on a Windows System. Note that this applies only to Windows operating systems. Registry will only be available if you are editing a VM Control Policy.

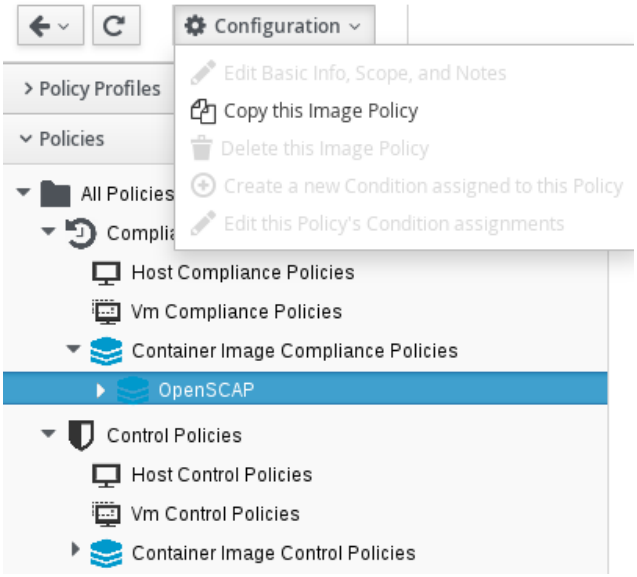


5. Click **Commit Expression Element Changes** to add the scope.
6. In the **Notes** area, make the required changes.
7. Click **Save**.

Copying a Policy

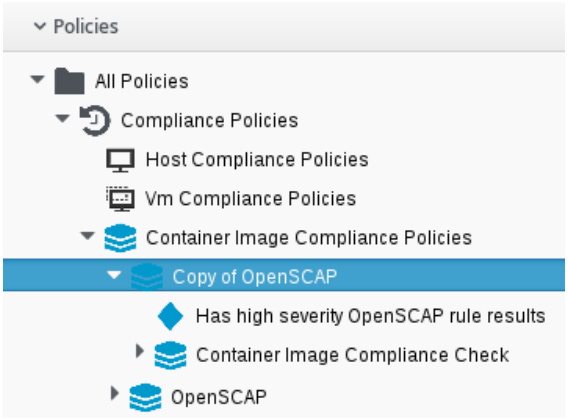
You can copy a policy if its contents are similar to a new one that you want to create, then change the condition or event associated with it. This enables you to make new policies efficiently.

1. Navigate to menu: **Control > Explorer**.
2. Click the **Policies** accordion, and select the policy that you want to copy.
3. Click **Configuration**, and an option to copy the policy appears; for example, **Copy this Image Policy**.



4. Click **OK** to confirm.

The new policy is created with a prefix of **Copy of** in its description, and it can be viewed in the Policy accordion.



Deleting a Policy

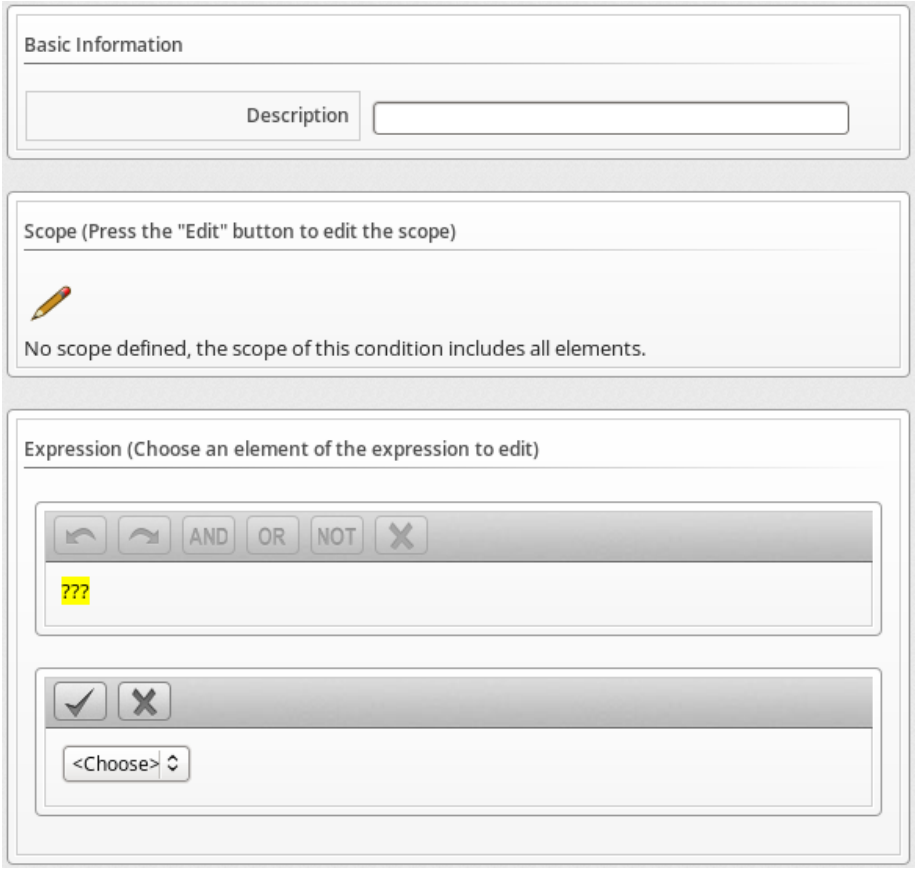
You can remove policies that you no longer need. You can remove only the policies that aren't assigned to a policy profile.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, and select the policy that you want to remove.
3. Click **Configuration**, **Delete this Host/VM and Instance/Replicator/Pod/Node/Image Policy**.
4. Click **OK** to confirm.

Creating a New Policy Condition

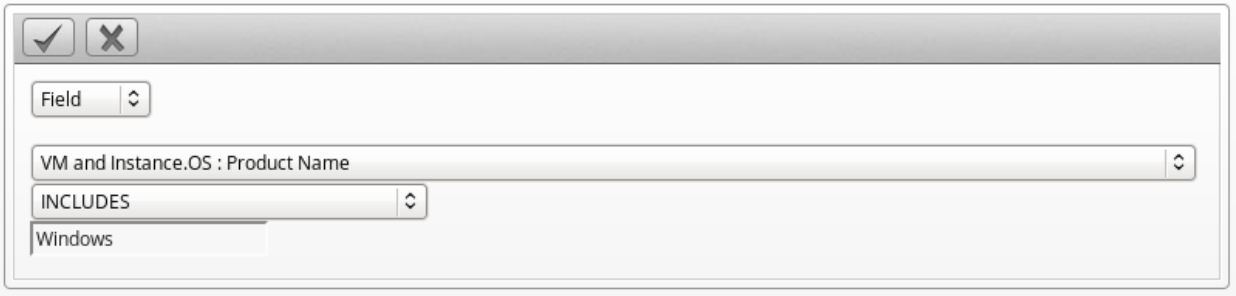
If you have not created a condition to use with this policy, you can create one directly from inside the policy. A condition can contain two elements: a scope, and an expression. The expression is mandatory, but the scope is optional. A scope is a general attribute that is quickly checked before evaluating a more complex expression. You can create a scope at either the policy or condition level.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, and select the policy that you want to create a new condition for.
3. Click **Configuration**, **Create a new Condition assigned to this Policy**.
4. Type in a **Description** for the condition. It must be unique to all the conditions.



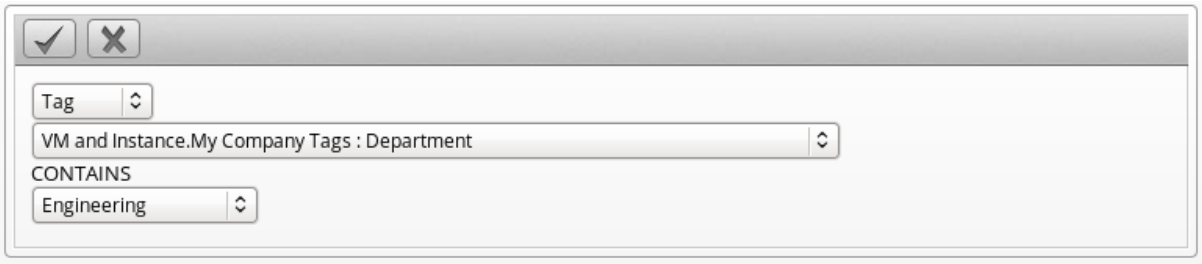
5. Click **Edit this Scope** in the **Scope** area to create a general expression based on a simple attribute, such as operating system version. Based on what you choose, different options display. **Scope** is optional.

    - Click **Field** to create criteria based on field values.

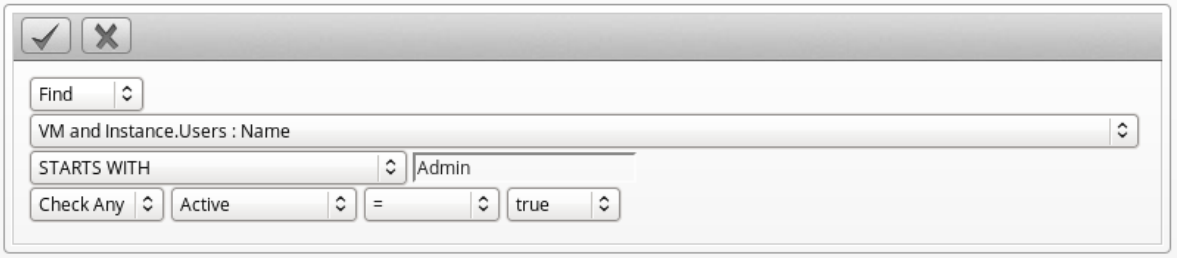


    - Click **Count of** to create criteria based on the count of something, such as the number of snapshots for a virtual machine, or the number of virtual machines on a host.

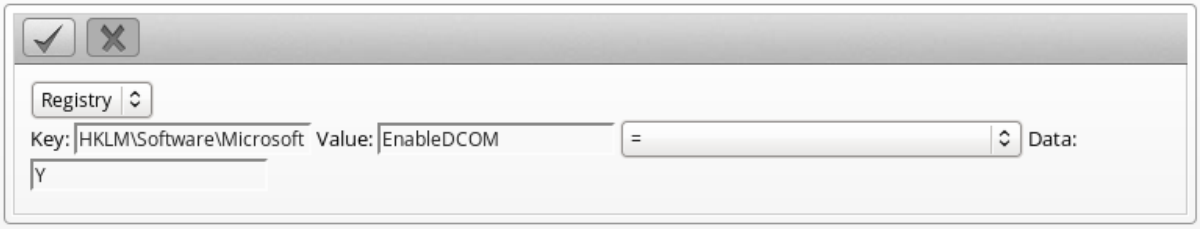
    - Click **Tag** to create criteria based on tags that are assigned to your resources. For example, you can check the power state of a virtual machine or see whether it is tagged as production.



- Click **Find** to seek a particular value, and then check a property. For example, finding the Admin account and checking that it is enabled. Use the following check commands:
  - **Check Any**: The result is true if one or more of the find results satisfy the check condition.
  - **Check All**: All of the find results must match for a true result.
  - **Check Count**: If the result satisfies the expression in check count, the result is true.



- Click **Registry** to create criteria based on registry values. For example, you can check whether DCOM is enabled on a Windows System. This applies only to Windows operating systems. Registry is only available if you are creating a VM Control Policy.



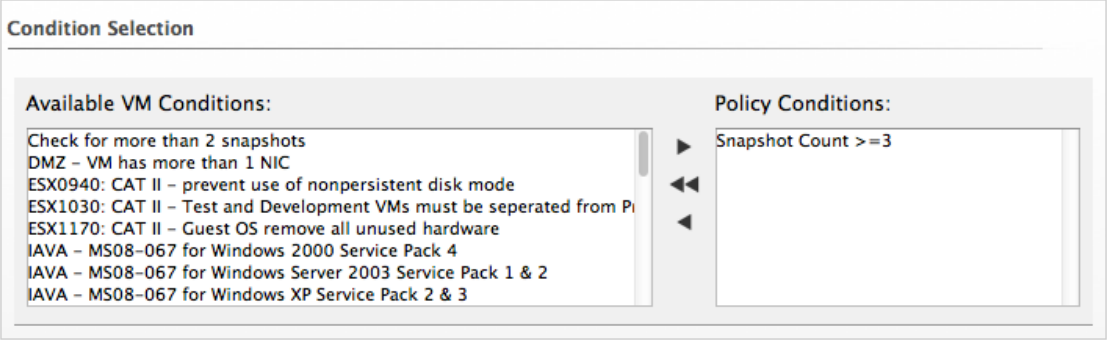
6. Click **Commit expression element changes** to add the scope.
7. Click **Edit this Expression** in the **Expression** area. Based on what you choose, options display according to the choices that are presented in the **Scope** area.
8. Click **Commit Expression Element Changes** to add the expression.
9. In **Notes**, type in a detailed explanation of the condition.
10. Click **Add**.

The condition is created and is assigned directly to the policy. **Note**: The condition can be assigned to other policies.

Editing Policy Condition Assignments

Use this procedure to use a condition that was created either separately or as part of another policy. You can also remove a condition from a policy that no longer applies.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, and select the policy that you want to assign conditions to.
3. Click **Configuration**, **Edit this Policy's Condition assignments**.
4. From the **Condition Selection** area, you can assign conditions to the policy, remove all conditions from the policy, or remove specific conditions from the policy.



- To add one or several conditions, select all the conditions that you want to apply from the **Available Conditions** box. Use `Ctrl` to add multiple conditions to a policy. Then, click **Move selected Conditions into this Policy**.
- Click **Remove all Conditions from this Policy** to unassign any conditions from this policy.
- To remove one or some conditions, select all the conditions that you want to remove from the **Policy Conditions** box. Use `Ctrl` to select multiple conditions. Then, click **Remove selected Conditions from this Policy**.

5. Click **Save**.

Editing Policy Event Assignments

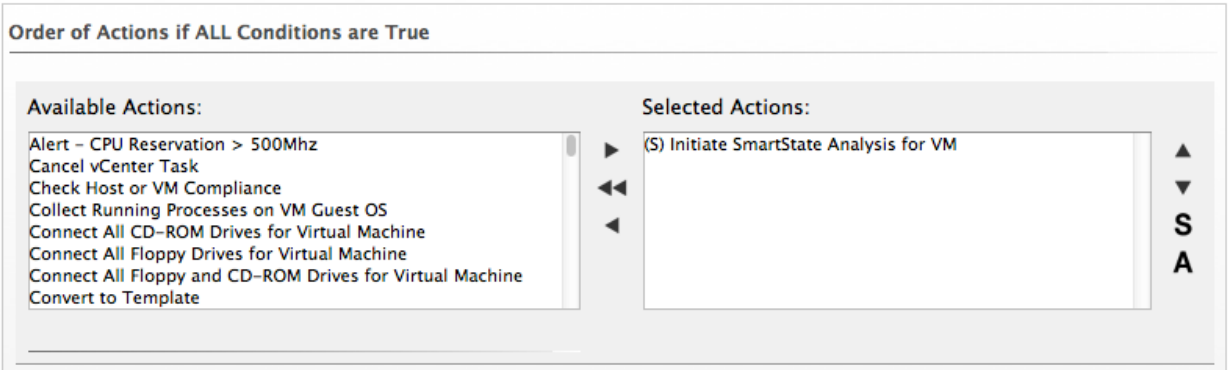
The policy evaluates its scopes and conditions when specified events occur in your virtual infrastructure. This procedure helps you to select those events and the actions that occur based on the evaluation of the scopes and conditions for the policy.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion and select the control policy that you want to assign events to.
3. Click **Configuration**, **Edit this Policy's Event assignments**.
4. Check all the events that you want to assign to this policy.
5. Click **Save**.

Assigning an Action to an Event

This procedure describes how to assign an action to an event.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, and select the policy that you want to assign actions to.
3. From the **Events** area, click the description of the event you want to assign an action to.
4. Click **Configuration**, **Edit Actions for this Policy Event**.
5. Select all the appropriate actions from the **Available Actions** box, inside the **Order of Actions if ALL Conditions are True**. These are the actions that take place if the resources meet the Condition of the Policy.



Note:

Each selected action can be executed synchronously or asynchronously; synchronous actions will not start until the previous synchronous action is completed, and asynchronous action allows the next action to start whether the first action has completed or not. Also, at least one Infrastructure Automation server in the Infrastructure Automation zone must have the notifier server role that is enabled for the trap to be sent.

6. Click the add button ›, then:
   - Click the action, then click A **Set selected Actions to Asynchronous** to make it asynchronous.
   - Click the action, then click S **Set selected Actions to Synchronous** to make it synchronous. If creating a synchronous action, use the up and down arrows to identify in what order you want the actions to run.
7. Select all the actions from the appropriate **Available Actions** box, inside of the **Order of Actions if ANY Conditions are False**. These are the actions that take place if the resources do not meet the condition of the policy.
8. Click **Save**.

Compliance Policies

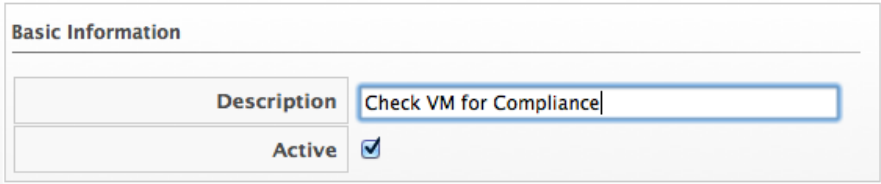
Compliance policies are designed to secure your environment by checking conditions that you create. These conditions can include the same conditions that you would use in a control policy, and most of the procedures are the same. However, a compliance policy automatically assigns the mark as a compliant action when the entity type (virtual machine or host, for example) to which the policy applies passes all of the conditions. If any of the conditions are not met, then the virtual machine or host is marked as non-compliant. The compliance status is shown in the summary screen for the entity type and on the compare and drift screens.

Creating a Compliance Policy

Create compliance policies by assigning or creating a condition. Infrastructure Automation automatically assigns the events and actions to the compliance policy as opposed to a control policy where you must define this yourself. The entity type (VM or host, for example) compliance check event is assigned to the compliance policy. A compliance policy runs the mark as compliant action when the virtual machine or host passes all of the conditions. If any of the conditions are not met, then the virtual machine or host is marked as non-compliant.

Carefully plan the purpose of your policy before creating it. You can also use a scope expression that is tested immediately when the compliance check event triggers the policy. If the item is out of scope, then the policy does not continue on to the conditions, and none of the associated actions run.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, and select **Compliance Policies**.
3. Select either **Host Compliance Policies** or **VM Compliance Policies** or **Replicator Compliance Policies** or **Pod Compliance Policies** or **Container Node Compliance Policies** or **Container Image Compliance Policies**.
4. Click **Configuration**, **Add a new Compliance Policy**.
5. Type in a **Description** for the policy.



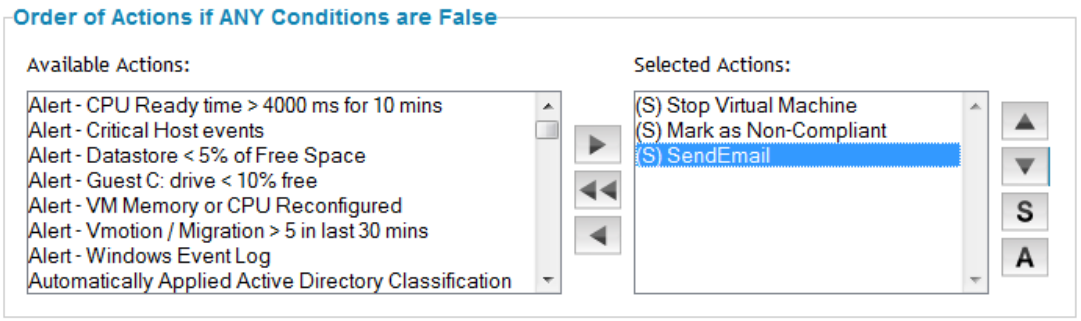
6. Uncheck **Active** if you do not want this policy to be processed even when assigned to a resource.
7. You can enter a scope here. (You can also create a scope as part of a condition, or not use one at all.) If the host or virtual machine is not included in the scope, no actions run.
8. In the **Notes** area, add a detailed explanation of the policy.
9. Click **Add**.

You must add one or several conditions:

- You can create a new condition by clicking **Configuration**, **Create a new Condition assigned to this Policy**, as described in Creating a New Policy Condition.
- You can use an existing condition by clicking **Configuration**, **Edit this Policy's Condition assignments**, as described in Editing Policy Condition Assignments.

By default, if any of the conditions are false, the virtual machine is marked as non-compliant. To add other actions, such as sending an email if the virtual machine fails the compliance test:

1. Click the **Compliance Check** event under the policy (exact name depends on entity type, for example **VM Compliance Check**.
2. Click **Configuration**, **Edit Actions for this Policy Event**.
3. Select **Stop Virtual Machine** and **Send Email** from the **Available Actions** area in **Order of Actions if ANY conditions are False**. (**Mark as Non-Compliant** should already be selected.)



4. Click **Move selected Actions into this Event**.
5. Click **Add**.

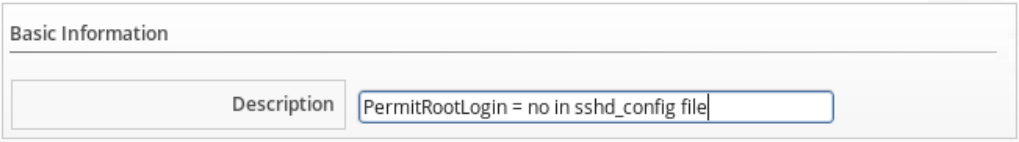
You can now make this part of a policy profile. After assigning the policy profile to the virtual machine, you can check it for its compliance status either on a schedule or on demand.

Creating a Compliance Condition to Check Host File Contents

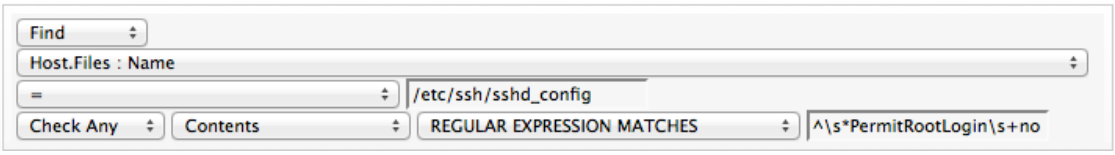
Infrastructure Automation Control provides the ability to create a compliance condition that checks file contents. Use this to be sure that internal operating system settings meet your security criteria. Regular expressions are used to create the search pattern. Test your regular expressions thoroughly before using them in a production environment.

Note that to search file contents you need to have collected the file using a host analysis profile.

1. Browse to menu: **Control > Explorer**.
2. Click the **Conditions** accordion, and select **Host Conditions**.
3. Click **Configuration**, **Add a New Host Condition**.
4. In **Basic Information**, type in a **Description** for the condition.



5. Editing the **Scope** area is not necessary for this procedure. Skip editing any **Scope** conditions.
6. If the **Expression** area is not automatically opened, click **Edit this Expression**, then edit the condition area to create a general condition based on a simple attribute. Based on what you choose, different options appear.
   - Click **Find**, then **Host.Files : Name**, and the parameters to select the file that you want to check.
   - Click **Check Any**, **Contents**, **Regular Expression Matches**, and type the expression. For example, if you want to make sure that permit root login is set to no, type `^\s*PermitRootLogin\s+no`.



7. Click **Commit expression element changes** to add the expression.

8. In **Notes** area, type in a detailed explanation of the condition.
9. Click **Add**.

Checking for Compliance

After you have created your compliance policies and assigned them to a policy profile, you can check compliance in two ways. You can either schedule the compliance check or perform the check directly from the summary screen.

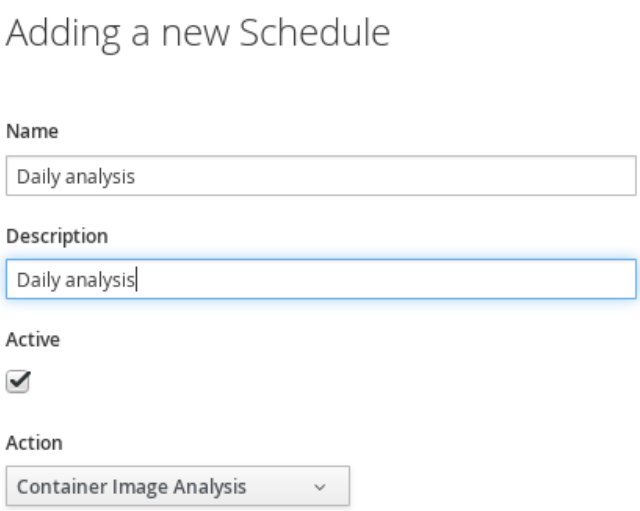
The compliance check runs all compliance policies that are assigned to the host or virtual machine. If the item fails any of the checks, it is marked as non-compliant in the item's summary screen.

Note:

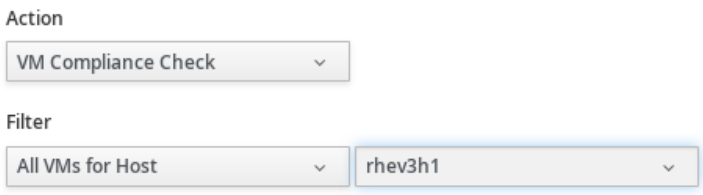
To schedule, you must have **`EvmRole-administrator`** access to the Infrastructure Automation server.

Scheduling a Compliance Check

1. Click **Configuration**.
2. Click the **Settings** accordion, and select **Schedules**.
3. Click **Configuration**, **Add a new Schedule**.
4. In the **Adding a new Schedule** area, type in a name and description for the schedule.



5. Select **Active** if you want to enable this scan.
6. From the **Action** dropdown, select the type of compliance check you want to schedule. Depending on the type of analysis you choose, you are presented with one of the following group boxes:
   - If you choose **VM Compliance Check**, you are presented with **VM Selection** where you can choose to check all VMs, all VMs for a specific provider, all VMs for a cluster, all VMs for a specific host, a single VM, or you can select VMs by using a global filter.



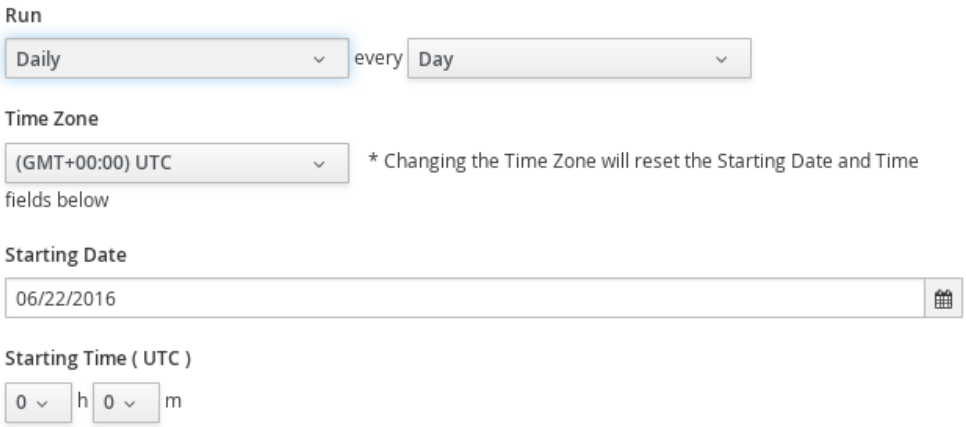
   - If you choose **Host Compliance Check**, you are presented with **Host Selection** where you can choose to analyze all hosts, all hosts for a specific provider, all hosts for a cluster, a single host, or you can select hosts by using a global filter.
   - If you choose **Container Image Compliance Check**, you are presented with **Image Selection** where you can choose to analyze all images, all images for a specific provider, or a single image.

Note:

You can schedule a host analysis only for connected virtual machines, not repository virtual machines that were discovered through that host. Since repository virtual machines do not retain a relationship with the host that discovered them, there is no current way to scan them through the

scheduling feature. The host is shown because it might have connected virtual machines in the future when the schedule is set to run.

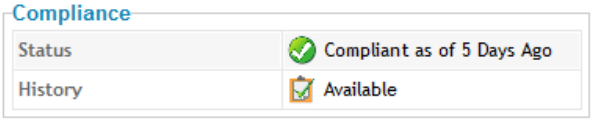
1. From the **Run** menu, select how often you want the analysis to run. Your options after that depend on which run option you choose.



   - Select **Once** to have the analysis run just one time.
   - Select **Hourly** to run the analysis hourly. You're prompted to select how many hours you want between each analysis.
   - Select **Daily** to run the analysis daily. You're prompted to select how many days you want between each analysis.
   - Select **Weekly** to run the analysis weekly. You're prompted to select how many weeks you want between each analysis.
   - Select **Monthly** to run the analysis monthly. You're prompted to select how many months you want between each analysis.
2. Select the time zone for the schedule.
3. Type or select a date to begin the schedule in **Starting Date**.
4. Select a starting time based on a 24-hour clock in the selected time zone.
5. Click **Add**.

Checking a Virtual Machine for Compliance from the Summary Screen

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**, select the virtual machine that you want to check for compliance.
2. Click **Policy**, and then **Check Compliance of Last Known Configuration**.
3. A confirmation message appears. Click **OK**.
4. To view the compliance history, click the virtual machine. Under **Compliance**, if **History** is **Available**, click it to see its compliance history.



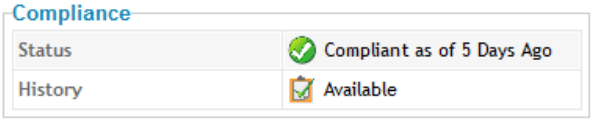
Checking a Host for Compliance from the Summary Screen

1. Browse to menu: **Compute > Infrastructure > Hosts**, click the host that you want to check for compliance.
2. Click **Policy**, and then **Check Compliance of Last Known Configuration** or **Analyze then Check Compliance**.
3. To view the compliance history, click **Available** next to **History**.

Checking a Replicator for Compliance from the Summary Screen

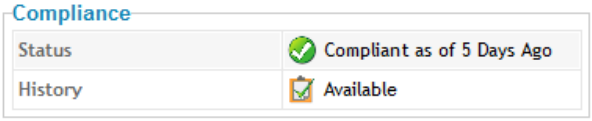
1. Browse to menu: **Compute > Containers > Replicators**, select the replicator that you want to check for compliance.

2. Click **Policy**, and then **Check Compliance of Last Known Configuration**.
3. A confirmation message appears. Click **OK**.
4. . To view the compliance history, click the replicator. Under **Compliance**, if **History** is **Available**, click to see its compliance history.



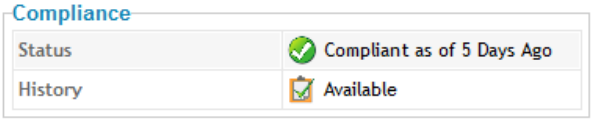
Checking a Pod for Compliance from the Summary Screen

1. Browse to menu: **Compute > Containers > Pods**, select the pod that you want to check for compliance.
2. Click **Policy**, and then **Check Compliance of Last Known Configuration**.
3. A confirmation message appears. Click **OK**.
4. To view the compliance history, click the pod. Under **Compliance**, if **History** is **Available**, click to see its compliance history.



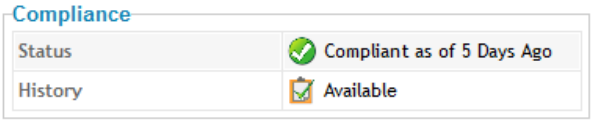
Checking a Container Node for Compliance from the Summary Screen

1. Browse to menu: Compute > Containers > Container Nodes**, click the node that you want to check for compliance.
2. Click **Policy**, and then **Check Compliance of Last Known Configuration**.
3. A confirmation message appears. Click **OK**.
4. To view the compliance history, click the node. Under **Compliance**, if **History** is **Available**, click to see its compliance history.



Checking a Container Image for Compliance from the Summary Screen

1. Browse to menu: **Compute > Infrastructure > Container Images**, select the container image that you want to check for compliance.
2. Click **Policy**, and then **Check Compliance of Last Known Configuration**.
3. A confirmation message appears. Click **OK**.
4. To view the compliance history, click the container image. Under **Compliance**, if **History** is **Available**, click to see its compliance history.



Conditions

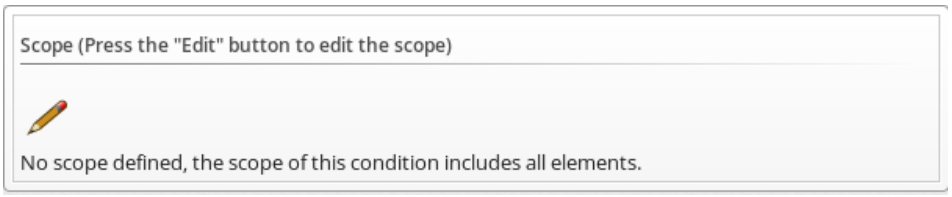
Conditions are tests that are performed on attributes of virtual machines. A condition can contain two elements, a scope, and an expression. The expression is mandatory, but the scope is optional. A scope is a general attribute that is quickly checked before evaluating a more complex expression. For example, you might use a scope to check the operating system, and use an expression to check for a specific set of applications or security patches that only apply to the operating system referenced in the scope. If no conditions, scope, or expression, are defined for a policy, the policy is considered unconditional and returns a true value.

Creating a Condition

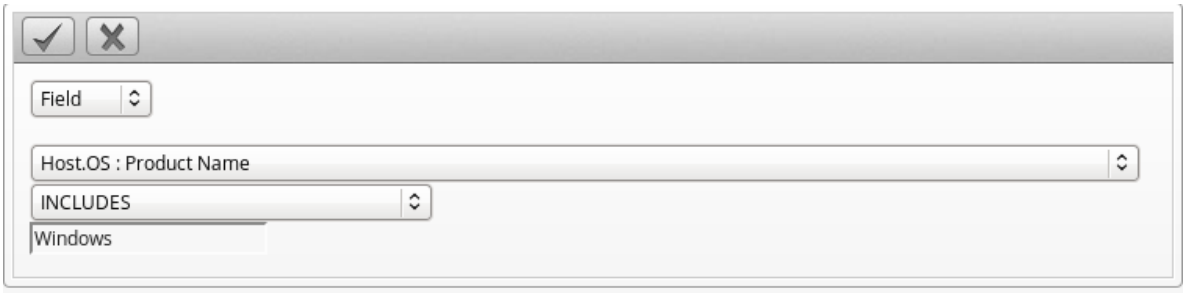
You can create a condition either from within a policy screen or by going directly to the expression editor in the Infrastructure Automation console. You need to define a description and an expression element. The expression element defines what criteria you want to use to test the condition.

1. Browse to menu: **Control > Explorer**.
2. Click the **Conditions** accordion, and select either **Host / Node Conditions** or **VM and Instance Conditions** or **Replicator Conditions** or **Pod** or **Node Conditions** or **Image Conditions**.
3. Click **Configuration**, then **Add a New Host / VM / Replicator / Pod / Node / Image Condition**.
4. Enter a **Description** for the condition.

5. Click **Edit this Scope** in the **Scope** area to create a general condition based on a simple attribute. Based on what you choose, different options appear. Creating a scope is optional.

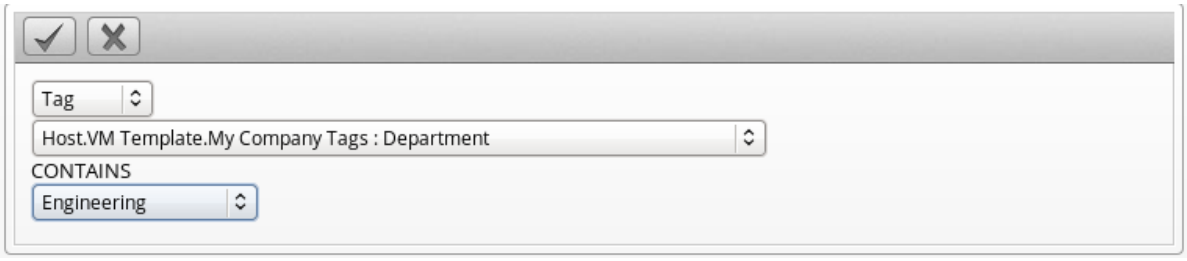


- Click **Field** to create criteria based on field values.



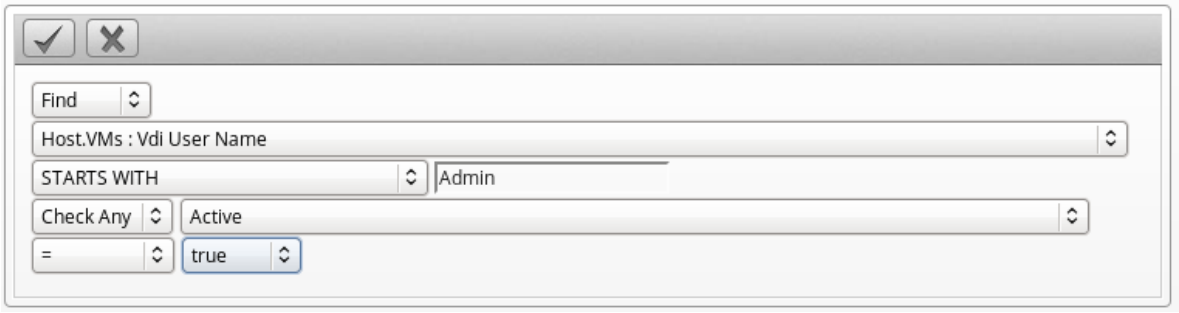
- Click **Count of** to create criteria based on the count of something, such as the number of network adapters on the host.

- Click **Tag** to create criteria based on tags that are assigned to your resources. For example, you can check the power state of a virtual machine or see whether it is tagged as production.

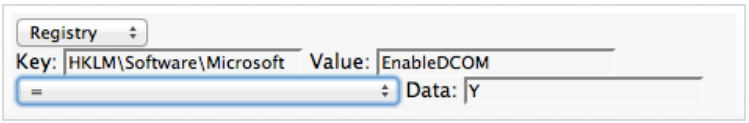


- Click **Find** to seek a particular value, and then check a property. For example, finding the Admin account and checking that it is enabled. Use the following check commands:
  - **Check Any**: The result is true if one or more of the find results satisfy the check condition.

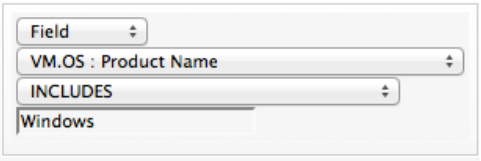
- **Check All**: All of the find results must match for a true result.
- **Check Count**: If the result satisfies the expression in check count, the result is true.



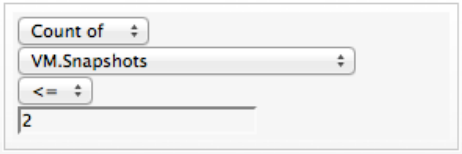
- Click **Registry** to create criteria based on registry values. For example, you can check whether DCOM is enabled on a Windows System. This applies only to Windows operating systems. Registry will only be available if you are creating a VM Condition.



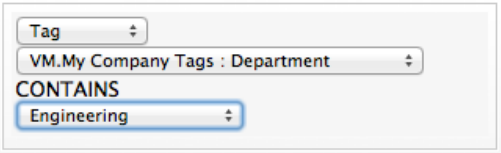
6. Click ✓ **Commit expression element changes** to add the scope.
7. Click **Edit this Expression** in the **Expression** area to create a general condition based on a simple attribute. Based on what you choose, different options appear.
   - Click **Field** to create criteria based on field values.



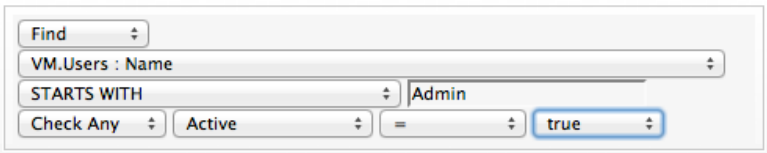
   - Click **Count of** to create criteria based on the count of something, such as the number of snapshots for a virtual machine, or the number of virtual machines on a host.



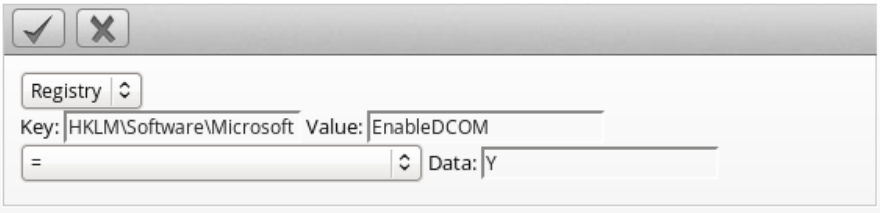
   - Click **Tag** to create criteria based on tags assigned to your resources. For example, you can check the power state of a virtual machine or see whether it is tagged as production.



   - Click **Find** to seek a particular value, and then check a property. For example, finding the Admin account and checking that it is enabled. Use the following check commands.
     - **Check Any**: The result is true if one or more of the find results satisfy the check condition.
     - **Check All**: All of the find results must match for a true result.
     - **Check Count**: If the result satisfies the expression in check count, the result is true.



- Click **Registry** to create criteria based on registry values. For example, you can check whether DCOM is enabled on a Windows System. This applies only to Windows operating systems.

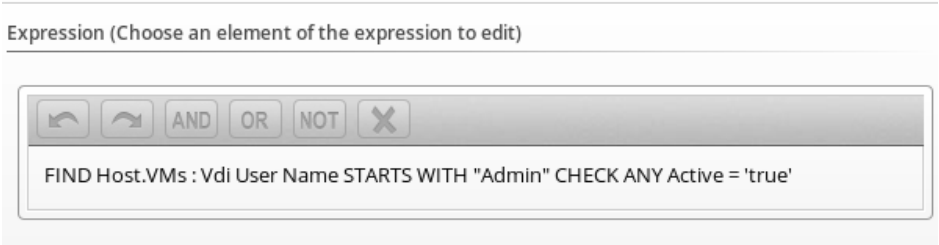


8. Click **Commit expression element changes** to add the expression.
9. In **Notes**, type in a detailed explanation of the condition.
10. Click **Add**.

Editing a Condition

Edit a condition to add more expressions to it or modify its properties. You can edit conditions that you created.

1. Browse to menu: **Control > Explorer**.
2. Click the **Conditions** accordion, and click the condition that you want to edit.
3. Click **Configuration**, **Edit this Condition**.
4. Click in either the **Scope** or **Expression** area, and click the part of the condition to edit.



5. Make the edits for the current expression.
   - Click **Commit expression element changes** to add the changes.
   - Click **Undo the previous change** to cancel the last action executed.
   - Click **Redo the previous change** to repeat the previous action executed.
   - Click **AND with a new expression element** to create a logical AND with a new expression element.
   - Click **OR with a new expression element** to create a logical OR with a new expression element.
   - Click **Wrap this expression element with a NOT** to create a logical NOT on an expression element.
   - Click **Remove this expression element** to take out the current expression element.
6. When you complete all of the changes to the condition, click **Save**.

Copying a Condition

You can copy a condition to create a similar condition, then change the values that are associated with it. You can copy the sample conditions that are provided to customize them to your environment.

1. Browse to menu: **Control > Explorer**.
2. Click the **Conditions** accordion, and select the condition that you want to copy.
3. Click **Configuration**, **Copy this Condition to a new Condition**.
4. Make all the changes that you need for the new condition. The description must be unique to all conditions.

5. Click **Add**.

Deleting a Condition

Remove conditions that are no longer applicable. You can delete only the conditions that are not part of a policy. To be able to delete the condition, you must remove the policy first.

1. Browse to menu: **Control > Explorer**.
2. Click the **Conditions** accordion, and click the condition that you want to remove.
3. Click **Configuration**, **Delete this VM and Instance Condition**.
4. Click **OK** to confirm.

Actions

Actions are performed after the condition is evaluated. Control comes with a set of default actions that you can choose from. You can also create some of your own.

Action	Description
Cancel vCenter Task	Stop current vCenter Task. Due to limitations of vCenter, this applies only to cloning tasks.
Check Host or VM Compliance	Run compliance checks.
Collect Running Processes on VM Guest OS	Collect the list of running processes from the guest operating system.
Connect All CD-ROM Drives for Virtual Machine	Connect all the CD-ROM drives for the virtual machine.
Connect All Floppy Drives for Virtual Machine	Connect all the floppy drives for the virtual machine.
Connect All Floppy and CD-ROM Drives for Virtual Machine	Connect all of the floppy and CD-ROM drives for virtual machine.
Convert to Template	Convert this virtual machine to a template.
Delete all Snapshots	Remove all snapshots for a virtual machine.
Delete Most Recent Snapshot	Removes a virtual machine's most recent snapshot.
Delete VM from Disk	Remove the virtual machine from disk.
Disconnect All CD-ROM Drives for Virtual Machine	Disconnect all the CD-ROM drives for the virtual machine.
Disconnect All Floppy Drives for Virtual Machine	Disconnect all the floppy drives for the virtual machine.
Disconnect All Floppy and CD-ROM Drives for Virtual Machine	Disconnect all of the floppy and CD-ROM drives for virtual machine.
Execute an external script	Run an external script.
Generate Audit Event	Write an entry to the audit log and to the VMDB.
Generate log message	Write an entry to the Infrastructure Automation log.
Initiate SmartState Analysis for Host	Start a SmartState Analysis for a host.
Initiate SmartState Analysis for VM	Start a SmartState Analysis for a virtual machine.
Invoke a Custom Automation	For use with Infrastructure Automation automate. It enables you to run tasks and notifications automatically.
Mark as Non-Compliant	Used with compliance policies. Mark resource as non-compliant. (Compliance status is viewable in summary screens.)
Prevent current event from proceeding	Stop the current event from continuing.
Put Virtual Machine Guest OS in Standby	Put the virtual machines operating system in standby mode.
Raise Automation Event	Used with Infrastructure Automation automate.
Refresh data from vCenter	Perform a refresh of the vCenter.
Remove Virtual Machine from Inventory	Take the virtual machine out of inventory.
Retire Virtual Machine	Retire the virtual machine. (It remains in inventory, but cannot be started.)
Show EVM Event on Timeline	To show the EVM event on the timeline.
Shutdown Virtual Machines Guest OS	Shut down the virtual machine's operating system.
Start Virtual Machine	Power on the virtual machine.
Stop Virtual Machine	Power off the virtual machine.
Suspend Virtual Machine	Suspend the virtual machine.

Default Actions and Descriptions

Custom Actions

You can create a custom action by using the Infrastructure Automation console. Enter a description and action type. Procedures for each type of action are shown in the following sections. When you create a policy, you can associate actions with specific events.

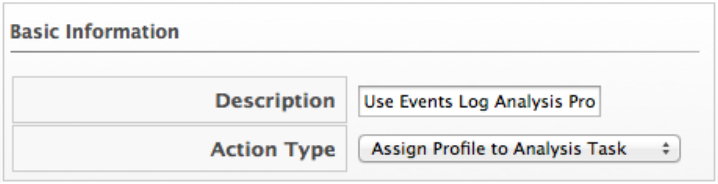
Custom Action	Description
Assign Profile to Analysis Task	When initiating a Smart State Analysis event, you can assign a specific analysis profile.
Create a Snapshot	Creates a snapshot with a name that you provide.
Delete Snapshots by Age	Removes snapshots based on how old they are.
Evaluate Alerts	Checks for alerts. This is required for the alert to be delivered.
Inherit Parent Tags	Assigns tags from the parent cluster, host, datastore, or resource pool.
Invoke a Custom Automation	For use with Infrastructure Automation automate.
Reconfigure CPUs	Reconfigure the number of CPUs for a virtual machine to the number you specify.
Reconfigure Memory	Reconfigure the amount of memory for a virtual machine to the amount you specify.
Remove Tags	Removes tags from the resource.
Run Ansible Playbook	Run an Ansible playbook against an inventory selection.
Send an E-mail	Send an email to an address that you provide. This type of action can be used in an alert.
Send an SNMP trap	Send an SNMP (Simple Network Management Protocol) trap to the host you specify. This type of action can be used for an alert.
Set a Custom Attribute in vCenter	Set the value of a custom attribute in vCenter.
Tag	Assign a company tag that you specify to a virtual machine.

Custom Actions and Descriptions

Creating an Assign Profile to Analysis Task Action

Use this action for assigning specific analysis profiles to virtual machines. You must create an analysis profile before assigning it to an action. You can only assign this action to an analysis start event.

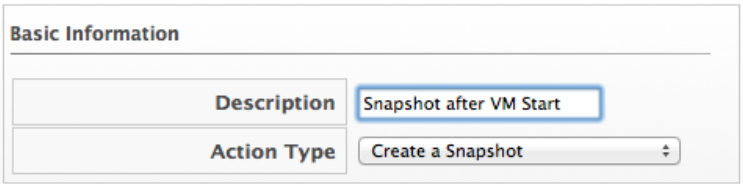
1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the **Action Type**.



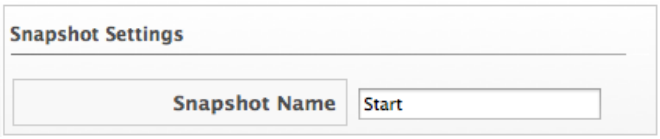
4. Select **Assign Profile to Analysis Task** from **Action Type**.
5. Select a profile from the **Analysis profiles**.
6. Click **Add**.

Creating a Snapshot Action

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



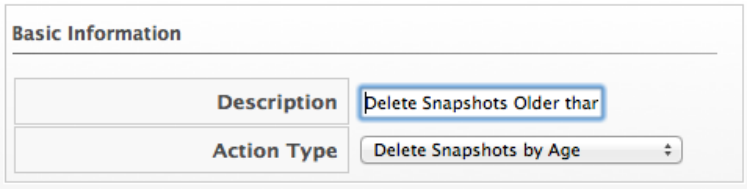
4. Select **Create a Snapshot** from **Action Type**.
5. Type in a **Snapshot Name**.



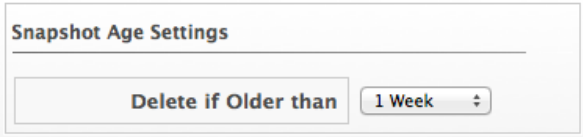
6. Click **Add** when you are finished.

Deleting Snapshots by Age

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



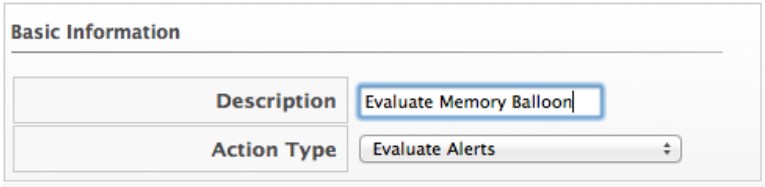
4. Select **Delete Snapshots by Age** from **Action Type**.
5. Select the age of snapshots to delete.



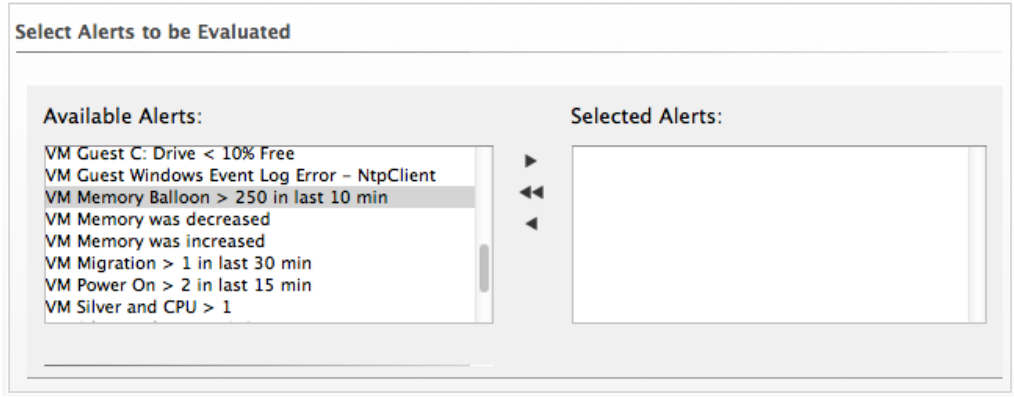
6. Click **Add**.

Evaluating an Alert

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



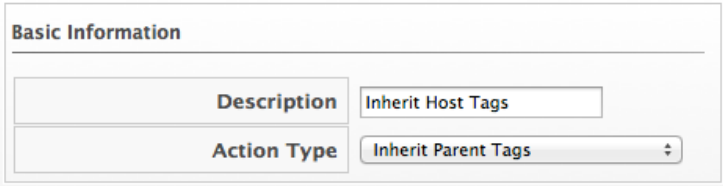
4. Select **Evaluate Alerts** from **Action Type**.
5. Select the alerts to be evaluated and click (Move selected Alerts into this Action). Use the `Ctrl` key to select multiple alerts.



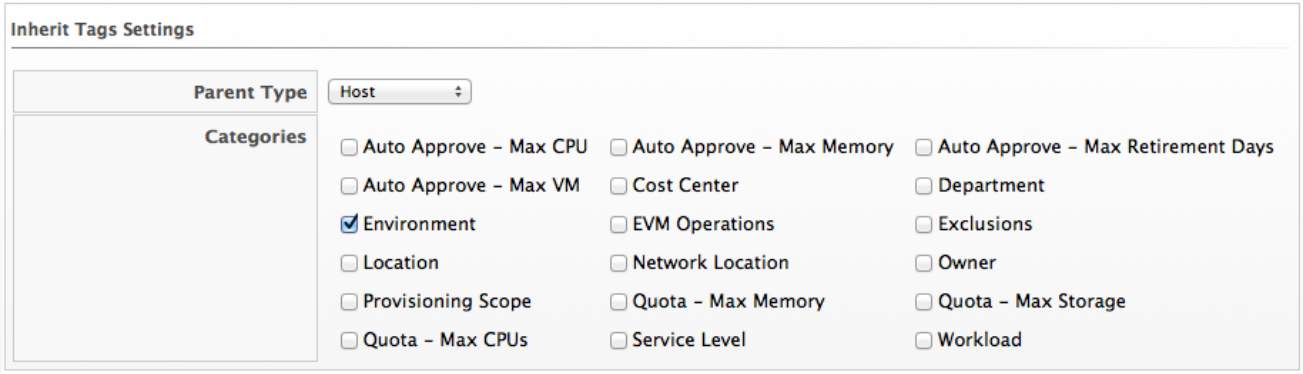
6. Click **Add**.

Creating an Inherit Tag Action

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, and click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



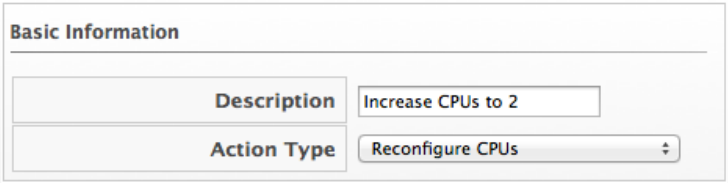
4. Select **Inherit Parent Tag** from **Action Type**.
5. Select the type of parent item to inherit from in **Parent Type**.
6. Check all categories that you want inherited.



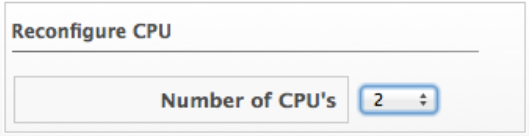
7. Click **Add**.

Creating a CPU Reconfigure Action

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



4. Select **Reconfigure CPUs** from **Action Type**.
5. Select a number from **Number of CPUs**.

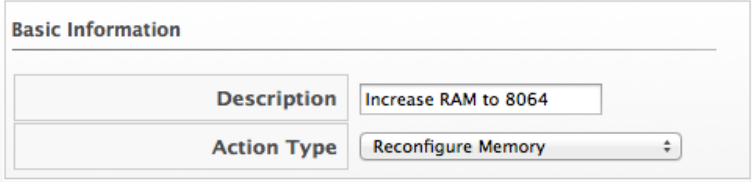


6. Click **Add**.

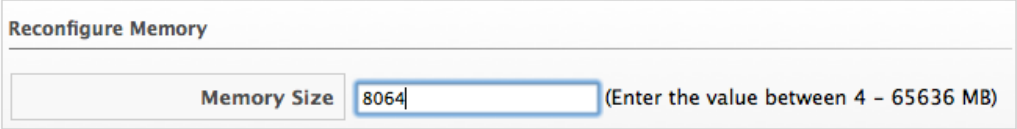
Creating a Memory Reconfigure Action

1. Browse to menu: **Control > Explorer**.

2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



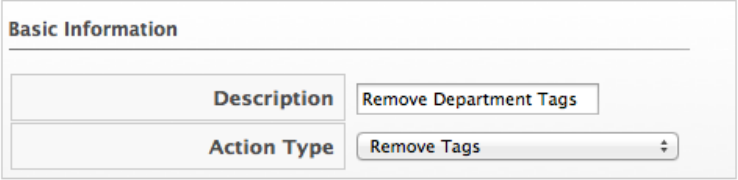
4. Select **Reconfigure Memory** from **Action Type**.
5. Type in a new value for **Memory Size**.



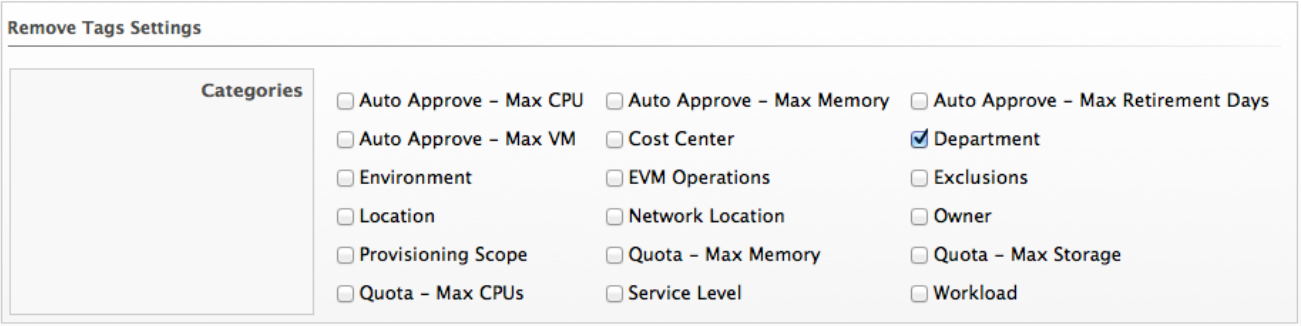
6. Click **Add**.

Creating a Remove Tag Action

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



4. Select **Remove Tags** from **Action Type**.
5. Check the category of tags you want to remove.



6. Click **Add**.

Creating an Ansible Playbook Run Action

Use this action to run an Ansible Playbook against your inventory. You must first sync a playbook repository and add an Ansible Playbook service catalog item.

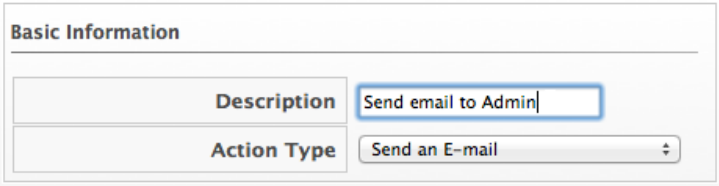
1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, and click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.
4. Select **Run Ansible Playbook** from **Action Type**.

5. Select the playbook catalog item to run from **Playbook Catalog Item**.
6. Check the inventory against which you run the Ansible playbook.
    - If **Specific Hosts** is selected, provide the IP or DNS names.
7. Click **Add**.

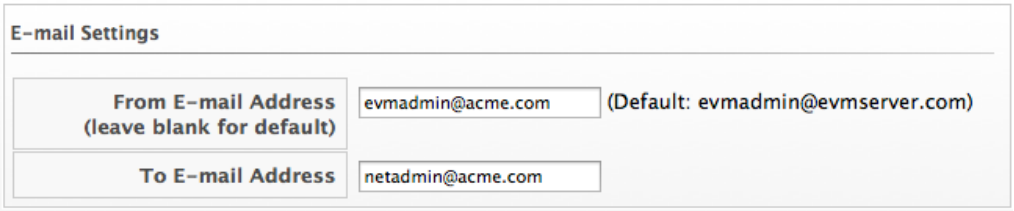
Creating an E-mail Action

To send emails from the Infrastructure Automation server, you must have the notifier server role enabled and have defined settings for SMTP email.

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



4. Select **Send an E-mail** from **Action Type**.
5. Type in a **From E-mail Address** and **To E-mail Address**.



6. Click **Add**.

Creating an SNMP Action

To send SNMP traps from the Infrastructure Automation server, you must have the **`Notifier`** server role and the SNMP daemons enabled.

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Enter a **Description** for the action.
4. Select **Send an SNMP** Trap from **Action Type**.
5. Type in the IP for the host to send the trap to, select the version of SNMP that you are using, and type in the Trap Object ID. Type in multiple hosts if you require the trap sent to multiple SNMP hosts.
    - If using SNMP V1, you are prompted for a Trap Number. Type 1, 2, or 3, based on the appropriate Suffix Number from table below.
    - If using SNMP V2, you are prompted for a Trap Object ID. Type info, warning, or critical, based on the table below.

Object ID	Suffix Number Added to PEN	PEN with the Suffix Added
info	1	1.3.6.1.4.1.33482.1
warn, warning	2	1.3.6.1.4.1.33482.2
crit, critical, error	3	1.3.6.1.4.1.33482.3

Trap Object ID and Suffix Number

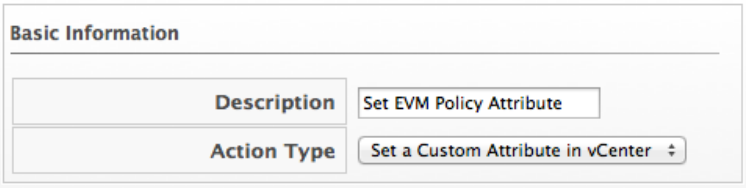
6. Type in the variables that you require in your message.
7. Click **Add**.

Note: When adding an SNMP action, be sure to set it as asynchronous.

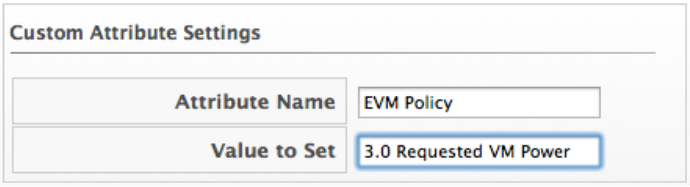
Creating a Set Custom Attribute Action

The custom attribute must already exist in vCenter. See vCenter documentation for instructions. In this example, an attribute called Infrastructure Automation policy already exists.

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



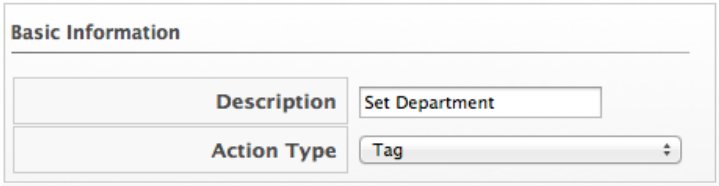
4. Select **Set a Custom Attribute in vCenter** from **Action Type**.
5. Type in the **Attribute Name** and **Value to Set**.



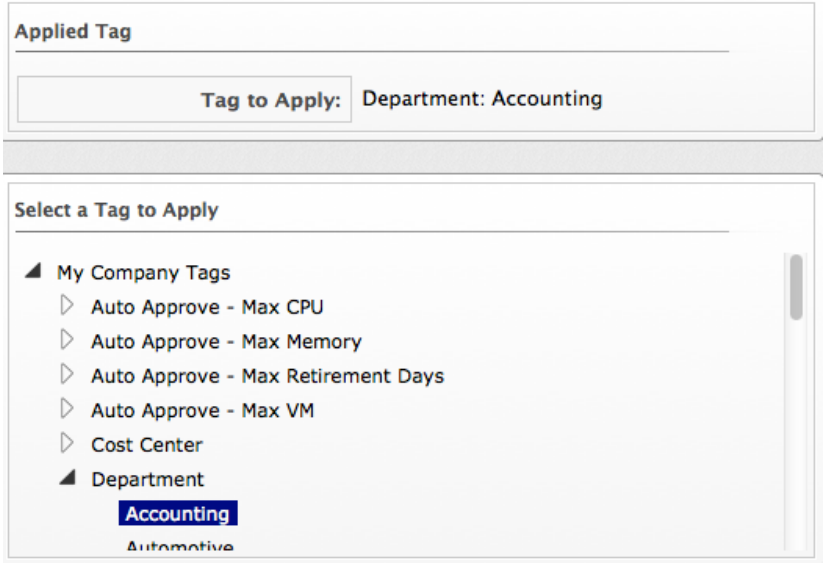
6. Click **Add**.

Creating a Tag Action

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, then click **Configuration**, **Add a new Action**.
3. Type in a **Description** for the action.



4. Select **Tag** from **Action Type**.
5. Click on the appropriate tag to apply from the list provided.



6. Click **Add**.

Editing an Action

Edit an action to modify its properties. You can't edit any of the default actions that are supplied with Infrastructure Automation. Only actions that you create can be changed.

Note: When you view an action, you can see what policies it is assigned to.

1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, and then click the action that you need to edit.
3. Click **Configuration**, **Edit this Action** on the detail view of the action.
4. Make any required changes.
5. Click **Save**.

The action is modified and can be added to a policy. If the action is already party of a policy, the policy is automatically updated.

Deleting an Action

Delete unused actions to keep your environment uncluttered. You can't delete default actions or actions that are currently assigned to a policy. The delete button is unavailable if the action is in use.

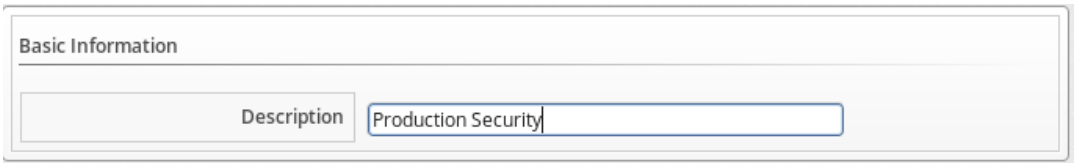
1. Browse to menu: **Control > Explorer**.
2. Click the **Actions** accordion, click the action that you need to remove.
3. Click **Configuration**, **Delete this Action** on the detail view of the tree.
4. Click **OK** to confirm.

Policy Profiles

Policy profiles are groups of policies that you can assign wholesale to virtual machines, providers, clusters, hosts, resource pools, replicators, pods, container nodes, and container images. Policy profiles provide a framework for easily managing and assigning different levels of security, across various types of cloud resources.

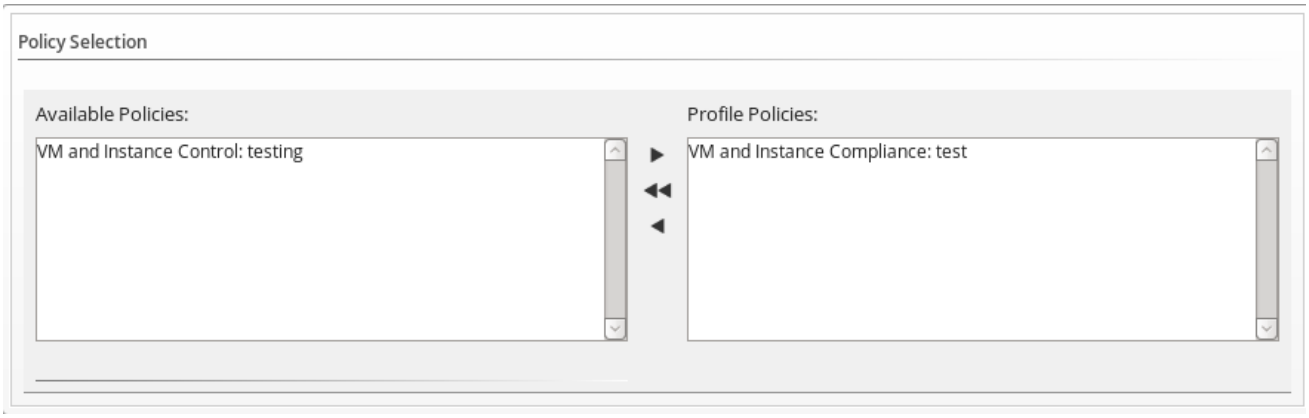
Creating Policy Profiles

1. Browse to menu: **Control > Explorer**.
2. Click on the **Policy Profiles** accordion, then click **Configuration**, then (**Add a New Policy Profile**).
3. In the **Basic Information** area, type in a unique description for the policy profile.



4. From **Available Policies** in the **Policy Selection** area select all the policies you need to apply to this policy profile. Use the `Ctrl` key to select multiple policies.

5. Click > to add the **Policies**.



6. Add to the **Notes** area if required.
7. Click **Add**.

The policy profile is added. You can now assign the policy profile to providers, hosts, and repositories. In addition, you can verify that the virtual machine complies with the policy profile by using the **Resultant Set of Policy** feature.

Deleting a Policy Profile

Remove policy profiles that you no longer need. This doesn't remove the policies that are associated with the policy profile.

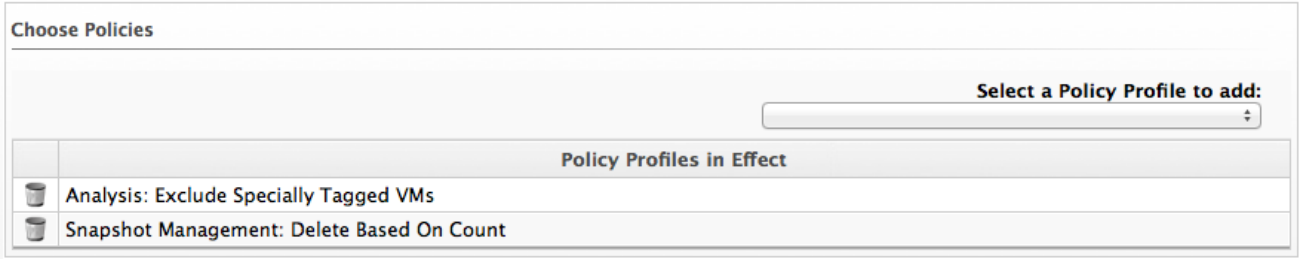
1. Browse to menu: **Control > Explorer**.
2. Click the **Policy Profile** accordion, and then click the policy profile that you want to remove.
3. Click **Configuration**, **Remove this Policy Profile**.
4. Click **OK** to confirm.

Simulating Policy

Before assigning a policy profile to a virtual machine, use the Infrastructure Automation controls policy simulation feature to determine if a virtual machine passes a policy profile.

Simulating Policy Profiles on Virtual Machines

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**, select the virtual machines that you need to evaluate.
2. Click **Policy**, and then click **Policy Simulation**.
3. From the **Select a Policy Profile to add** dropdown, click the policy that you need to apply to the selected virtual machines.



4. The virtual machine thumbnail displays in the **Policy Simulation** area.
    - A check sign in the virtual thumbnail shows that the virtual machine passes policy.
    - A minus sign in the virtual thumbnail shows that the virtual machine fails policy.
5. Click a virtual machine in the **Policy Simulation** area to see its details.
6. Expand a policy profile by clicking it to see its member policies and the status of the conditions.
    - Check **Show out of scope items** to show all conditions, whether the virtual machine passes the scope part of the condition. Uncheck it to hide conditions where the scope part fails.
    - Next to **Show policies**, check **Successful** to show policies that are passed and check **Failed** to see the policies that have failed. The default is to show both.
    - Items in green text passed the condition.
    - Items in red text failed the condition.
    - Items in red italics failed the condition, but do not change the outcome of the scope.

If you evaluate multiple policy profiles, you can see both policy profiles and a tree expanding down to their conditions.

Assigning Policy Profiles

After creating your policy profiles, you are ready to evaluate and assign them.

- Assign a policy profile to a virtual machine to apply the policy profile to a specific virtual machine, independent of its related host, provider, or repository.
- Assign a policy profile to a provider to apply the policy profile to all virtual machines, hosts, replicators, pods, container nodes or container images registered to that provider.
- Assign a policy profile to a replicator to apply the policy profile to that specific replicator.
- Assign a policy profile to a pod to apply the policy profile to that specific pod.
- Assign a policy profile to a container node to apply the policy profile to that specific node.
- Assign a policy profile to a container image to apply the policy profile to that specific image.
- Assign a policy profile to a cluster to apply the policy profile to all virtual machines or hosts assigned to that cluster.
- Assign a VM policy profile to a host to apply the policy profile to that specific host or all virtual machines registered to that host.
- Assign a VM policy profile to a resource pool to apply the policy profile to all virtual machines or hosts assigned to that resource pool.

Assigning Policy Profiles to an Infrastructure Provider

1. Browse to menu: **Compute > Infrastructure > Providers**, verify the provider you need to assign the policy profiles to.
2. Click (**Policy**), and then click (**Manage Policies**).
3. From the **Select Policy Profiles** area, you can click on the triangle next to a desired policy profile to expand it and see its member policies.
4. Check the policy profiles you require to apply to the provider. It turns blue to show its assignment state has changed.
5. Click **Save**.

Removing Policy Profiles from an Infrastructure Provider

1. Browse to menu: **Compute > Infrastructure > Providers**, check the providers that you want to remove the policy profile from.

2. Click **Policy**, and then click **Manage Policies**.
3. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
4. Click **Save**.

Assigning Policy Profiles to a Cluster

1. Browse to menu: **Compute > Infrastructure > Clusters**, check the clusters that you need to assign policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, you can click the triangle next to a desired policy profile to expand it and see its member policies.
4. Check the policy profiles that you need to apply to the cluster. It turns blue to show that its assignment state changed.
5. Click **Save**.

Removing Policy Profiles from a Cluster

1. Browse to menu: **Compute > Infrastructure > Clusters**, check the clusters that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, you can click the triangle next to a desired policy profile to expand it and see its member policies.
4. Uncheck the policy profiles that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to a Host

1. Browse to menu: **Compute > Infrastructure > Hosts**, check the hosts that you need to assign policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Check the policy profiles that you need to apply to the host. It turns blue to show that its assignment state changed.
5. Click **Save**.

Removing Policy Profiles from a Host

1. Browse to menu: **Compute > Infrastructure > Hosts**, check the hosts that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. Uncheck the policy profiles that you need to remove. It turns blue to show that its assignment state changed.
4. Click **Save**.

Assigning Policy Profiles to a Virtual Machine

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**, check the virtual machines that you need to assign policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Check the policy profiles that you need to apply to the host. It turns blue to show that its assignment state changed.
5. Click **Save**.

Removing Policy Profiles from a Virtual Machine

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**, check the virtual machines that you want to remove the policy profile from.
2. Click **Policy**, and then click **Manage Policies**.

3. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
4. Click **Save**.

Assigning Policy Profiles to a Resource Pool

1. Browse to menu: **Compute > Infrastructure > Resource Pools**, check the resource pools that you need to assign policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Click the policy profiles that you need to apply to the resource pools. It turns blue to show that its assignment state changed.
5. Click **Save**.

Removing Policy Profiles from a Resource Pool

1. Browse to menu: **Compute > Infrastructure > Resource Pools**, check the resource pools that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Uncheck the policy profiles that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to a Cloud Provider

1. Browse to menu: **Compute > Clouds > Providers** and check the provider that you need to assign the policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a wanted policy profile to expand it and see its member policies.
4. Check the policy profiles that you need to apply to the provider. The ones that are different from the previous setting shows in blue.
5. Click **Save**.

Removing Policy Profiles from a Cloud Provider

1. Browse to menu: **Compute > Clouds > Providers**, check the providers that you need to remove the policy profile from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to a Network Provider

1. Browse to menu: **Networks > Providers**, check the network provider that you need to assign the policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a wanted policy profile to expand it and see its member policies.
4. Check the policy profiles that you need to apply to the provider. The ones that are different from the previous setting shows in blue.
5. Click **Save**.

Removing Policy Profiles from a Network Provider

1. Browse to menu: **Networks > Providers**, check the network providers that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.

4. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to a Container Provider

1. Browse to menu: **Compute > Containers > Providers** and select the provider that you need to assign the policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a wanted policy profile to expand and see its member policies.
4. Select the policy profiles that you need to apply to the provider. It turns blue to show the selection.
5. Click **Save**.

Removing Policy Profiles from a Container Provider

1. Browse to menu: **Compute > Containers > Providers**, select the container providers that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to a Replicator

1. Browse to menu: **Compute > Containers > Replicators** and select the replicator that you need to assign the policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a wanted policy profile to expand and see its member policies.
4. Select the policy profiles that you need to apply to the replicator. It turns blue to show the selection.
5. Click **Save**.

Removing Policy Profiles from a Replicator

1. Browse to menu: **Compute > Containers > Replicators**, select the replicators that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to a Pod

1. Browse to menu: **Compute > Containers > Pods** and select the pod that you need to assign the policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a wanted policy profile to expand and see its member policies.
4. Select the policy profiles that you need to apply to the pod. It turns blue to show the selection.
5. Click **Save**.

Removing Policy Profiles from a Pod

1. Browse to menu: **Compute > Containers > Pods**, select the pods that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.

4. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to a Container Node

1. Browse to menu: **Compute > Containers > Container Nodes** and select the container node that you need to assign the policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a wanted policy profile to expand and see its member policies.
4. Select the policy profiles that you need to apply to the node. It turns blue to show the selection.
5. Click **Save**.

Removing Policy Profiles from a Container Node

1. Browse to menu: Compute > Containers > Container Nodes**, select the container nodes that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to a Container Image

1. Browse to menu: **Compute > Containers > Container Images** and select the image that you need to assign the policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a wanted policy profile to expand and see its member policies.
4. Select the policy profiles that you need to apply to the image. It turns blue to show the selection.
5. Click **Save**.

Removing Policy Profiles from a Container Image

1. Browse to menu: **Compute > Containers > Container Images**, select the container images that you need to remove the policy profiles from.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Assigning Policy Profiles to an Instance

1. Browse to menu: **Compute > Clouds > Instances**, check the instances that you want to assign policy profiles to.
2. Click **Policy**, and then click **Manage Policies**.
3. From the **Select Policy Profiles** area, click the triangle next to a wanted policy profile to expand it and see its member policies.
4. Check the policy profiles that you want to apply to the instances. It turns blue to show that its assignment state changed.
5. Click **Save**.

Removing Policy Profiles from an Instance

1. Browse to menu: **Compute > Clouds > Instances**, check the instances that you need to remove the policy profile from.
2. Click **Policy**, and then click **Manage Policies**.

3. From the **Select Policy Profiles** area, click the triangle next to a desired policy profile to expand it and see its member policies.
4. Uncheck the policy profile that you need to remove. It turns blue to show that its assignment state changed.
5. Click **Save**.

Disabling a Policy in a Policy Profile

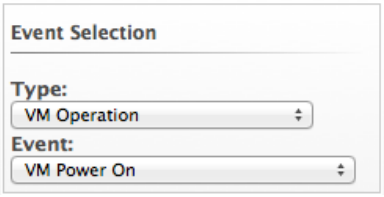
You can disable one policy in a profile without removing it from the policy, perhaps for trouble shooting purposes or because the policy is not required temporarily.

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, then navigate to the policy that you need to disable or navigate to the policy from the policy profile.
3. Click **Configuration**, **Edit Basic Info, Scope, Notes**.
4. Uncheck **Active**.
5. Click **Save**.

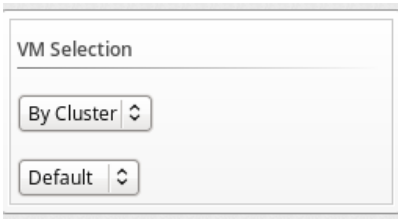
Viewing Policy Simulation - Resultant Set of Policy (RSOP)

After the Policy Profiles are assigned, you can see the final result of the resolution of all policies based on which Events occur. Based on the result, you can adjust your Policies. To view RSOP, go to the control area in the Infrastructure Automation console.

1. Browse to menu: **Control > Simulation**.
2. From the **Event Selection** area, select a type of event, and then the specific event you need the result for.



3. From the **VM Selection** area, select the virtual machine from a provider, cluster, host, or a single virtual machine.



4. Click **Submit**.

Exporting and Importing Analysis Profiles

Infrastructure Automation SmartState analysis requires an analysis profile to select the files to be scanned by a compliance policy. Infrastructure Automation allows the export and import of SmartState analysis profiles by using the command line and rake commands. As a result, approved configurations can be easily imported into customer environments, without having to manually recreate the profile through the user interface.

1. Change to the **vmdb** directory:

   ```
   cd /var/www/miq/vmdb
   ```

2. Create an export directory:

   ```
   $ mkdir exports
   ```

3. To export an analysis profile, run:

   ```
   bundle exec rake evm:export:scan_profiles -- --directory exports
   ```

4. To import the default analysis profile, run:

   ```
   bundle exec rake evm:import:scan_profiles -- --source exports/host_default.yaml
   ```

   Note:

If the default profile exists in Infrastructure Automation, the new profile does not overwrite the old profile. Instead, it duplicates the file items in the default profile.

Appendix

Events

Events are triggers that cause a condition to be tested. Control provides several Events, that can be divided into functional types. Events cannot be modified.

Category	Description
Container Operation	Events related to container analysis.
Datastore Operation	Events related to datastore analysis.
Authentication Validation	Events related to credential validation for hosts and providers.
Company Tag	Events related to assigning and removing company tags from an infrastructure object.
Compliance	Events related to checking compliance policies.
Host Operation	Events related to the connection state of a host and status of a SmartState Analysis on a host.
Orchestration Lifecycle	Events related to orchestration lifecycle, such as retirement.
Physical Server Operation	Events related to the connection state of a physical server.
VM Configuration	Events associated with a change in configuration of a virtual machine. These include, but are not limited to, clone, create, template create, and settings change.
VM Lifecycle	Events such as virtual machine discovery, provisioning, and virtual machine retirement.
VM Operation	Events associated with power states or locations of virtual machines and virtual desktop machines. These include, but are not limited to, power off, power on, reset, resume, shutdown, and suspend.
Service Lifecycle	Events associated with service lifecycle. These include, but are not limited to, provisioning completed, start request, started, stop request, stopped, retirement warning, and retired.

Event Types

Each type has a set of events that you can select to trigger the checking of a condition.

Event	Description
Container Image Analysis Request	Check the condition when an analysis of a container image is requested.
Container Image Analysis Complete	Check the condition when an analysis of a container image completes.
Container Image Discovered	Check the condition when a new container image is discovered.
Container Image Compliance Check	Check the condition when a compliance check is performed on an image.
Container Image Compliance Passed	Check the condition when an image passes a compliance check.
Container Image Compliance Failed	Check the condition when an image fails a compliance check.
Container Node Failed Mount	Check the condition when a node fails to mount a volume for a pod.
Container Node Invalid Disk Capacity	Check the condition when a node’s disk capacity is invalid.
Container Node Not Ready	Check the condition when a node is not ready.
Container Node Not Schedulable	Check the condition when a node is not schedulable.
Container Node Ready	Check the condition when a node is ready.

Event	Description
Container Node Schedulable	Check the condition when a node is schedulable.
Container Node Rebooted	Check the condition when a node reboots.
Container Node Compliance Check	Check the condition when a compliance check is performed on a node.
Container Node Compliance Passed	Check the condition when a node passes a compliance check.
Container Node Compliance Failed	Check the condition when a node fails a compliance check.
Pod Compliance Check	Check the condition when a compliance check is performed on a pod.
Pod Compliance Passed	Check the condition when a pod passes a compliance check.
Pod Compliance Failed	Check the condition when a pod fails a compliance check.
Pod Container Created	Check the condition when a container is created in a pod.
Pod Container Failed	Check the condition when a container in a pod fails.
Pod Container Killing	Check the condition when a container in a pod is killed.
Pod Container Started	Check the condition when a container in a pod is started.
Pod Container Stopped	Check the condition when a container in a pod is stopped.
Pod Container Unhealthy	Check the condition when a container in a pod is unhealthy.
Pod Deadline Exceeded	Check the condition when a pod with specified deadline exceeds it and is terminated.
Pod Failed Scheduling	Check the condition when scheduling a pod fails.
Pod Failed Sync	Check the condition when getting a pod to its desired state fails (a frequent reason is failure to download the image).
Pod Failed Validation	Check the condition when a pod validation fails.
Pod hostPort Conflict	Check the condition when a pod hostPort conflict occurs.
Pod Insufficient Free CPU	Check the condition when there is an insufficient free CPU in a pod.
Pod Insufficient Free Memory	Check the condition when there is an insufficient free memory in a pod.
Pod nodeSelector Mismatching	Check the condition when a pod nodeSelector mismatches.
Pod Out of Disk	Check the condition when a pod is out of disk space.
Pod Scheduled	Check the condition when a pod is scheduled onto the node.
Replicator Failed Creating Pod	Check the condition when a replicator fails creating a pod.
Replicator Successfully Created Pod	Check the condition when a replicator successfully creates a pod.
Replicator Compliance Check	Check the condition when a compliance check is performed on a replicator.
Replicator Compliance Passed	Check the condition when a replicator passes a compliance check.
Replicator Compliance Failed	Check the condition when a replicator fails a compliance check.
Database Failover Executed	Check the condition when a database failover is executed.
Datastore Analysis Complete	Check the condition when a SmartState Analysis of a datastore completes.

Event	Description
Datastore Analysis Request	Check the condition when a SmartState Analysis for a datastore is requested from the user interface.
Host Added to Cluster	Check the condition when a host is added to a cluster.
Host Analysis Complete	Check the condition when a SmartState Analysis of host completes.
Host Analysis Request	Check the condition when a SmartState Analysis is requested from the Infrastructure Automation console.
Host Auth Changed	Check the condition when host authentication credentials are changed in the Infrastructure Automation console.
Host Auth Error	Check the condition if there is any other error connecting to the host such as ssh/vim handshaking problems, timeouts, or any other uncategorized error.
Host Auth Incomplete Credentials	Check the condition if host authentication credentials are not complete in the user interface.
Host Auth Invalid	Check the condition if Infrastructure Automation is able to communicate with the host and the credentials fail.
Host Auth Unreachable	Check the condition if Infrastructure Automation is unable to communicate with the host.
Host Auth Valid	Check the condition when the host authentication credentials entered in the Infrastructure Automation console are valid.
Host C & U Processing Complete	Check the condition when the processing of capacity and utilization data has finished.
Host Compliance Check	Check the condition when a compliance check is performed on a host.
Host Compliance Failed	Check the condition when a host fails a compliance check.
Host Compliance Passed	Check the condition when a host passes a compliance check.
Host Connect	Check the condition when a host connects to a provider.
Host Disconnect	Check the condition when a host disconnects from a provider.
Host Maintenance Enter Request	Check the condition when a host requests to enter maintenance mode.
Host Maintenance Exit Request	Check the condition when a host requests to exit maintenance mode.
Host Provision Complete	Check the condition when the host provision is complete.
Host Reboot Request	Check the condition when someone tries to reboot a host from the Infrastructure Automation console.
Host Removed from Cluster	Check the condition when a host is removed from a cluster.
Host Reset Request	Check the condition when a host is restarted from the Infrastructure Automation console.
Host Shutdown Request	Check the condition when a host is shut down from the Infrastructure Automation console.
Host Standby Request	Check the condition when someone tries to put the operating system of a host in standby from the Infrastructure Automation console.
Host Start Request	Check the condition when a host is started from the Infrastructure Automation console.
Host Stop Request	Check the condition when a host is requested to stop from the Infrastructure Automation console.
Host Vmotion Disable Request	Check the condition when a request to disable vMotion on a host is created from the Infrastructure Automation console.
Host Vmotion Enable Request	Check the condition when a request to enable vMotion on a host is created from the Infrastructure Automation console.
Orchestration Stack Retire Request	Check the condition when an orchestration stack retirement request is created from Infrastructure Automation.
Physical Server Reset	Check the condition when a physical server is restarted from the Infrastructure Automation console.
Physical Server Shutdown	Check the condition when a physical server is shut down from the Infrastructure Automation console.
Physical Server Start	Check the condition when a physical server is started from the Infrastructure Automation console.
Provider Auth Changed	*For use only with Infrastructure Automation automate, for future use in policies.* Check the condition when provider authentication credentials are changed in the user interface.

Event	Description
Provider Auth Error	*For use only with Infrastructure Automation automate, for future use in policies.* Check the condition if there is any other error connecting to the provider such as ssh/vim handshaking problems, timeouts, or any other uncategorized error.
Provider Auth Incomplete Credentials	*For use only with Infrastructure Automation automate, for future use in policies.* Check the condition if provider authentication credentials are not complete in the Infrastructure Automation console.
Provider Auth Invalid	*For use only with Infrastructure Automation automate, for future use in policies.* Check the condition if Infrastructure Automation is able to communicate with the provider and the credentials fail.
Provider Auth Unreachable	*For use only with Infrastructure Automation automate, for future use in policies.* Check the condition if Infrastructure Automation is unable to communicate with the provider.
Provider Auth Valid	*For use only with Infrastructure Automation automate, for future use in policies.* Check the condition when the provider authentication credentials entered in the user interface are valid.
Provider Compliance Check	Check the condition when a compliance check is performed on a provider.
Provider Compliance Failed	Check the condition when a provider fails a compliance check.
Provider Compliance Passed	Check the condition when a provider passes a compliance check.
Service Provision Complete	Check the condition when the service provision is complete.
Service Retire Request	Check the condition when a service retirement request is created from Infrastructure Automation.
Service Retired	Check the condition when the service has been retired.
Service Retirement Warning	Check the condition when the service is about to retire.
Service Start Request	Check the condition when the service has been requested to start.
Service Started	Check the condition when the service has started.
Service Stop Request	Check the condition when the service has been requested to stop.
Service Stopped	Check the condition when the service has stopped.
Tag Complete	Check the condition after a company tag is assigned.
Tag Parent Cluster Complete	Check the condition after a company tag is assigned to a virtual machine's parent cluster.
Tag Parent Datastore Complete	Check the condition after a company tag is assigned to a virtual machine's parent datastore.
Tag Parent Host Complete	Check the condition after a company tag is assigned to a virtual machine's parent host.
Tag Parent Resource Pool Complete	Check the condition after a company tag is assigned to a virtual machine's parent resource pool.
Tag Request	Check the condition when assignment of a company tag is attempted.
Un-Tag Complete	Check the condition when a company tag is removed.
Un-Tag Parent Cluster Complete	Check the condition after a company tag is removed from a virtual machine's parent cluster.
Un-Tag Parent Datastore Complete	Check the condition after a company tag is removed from a virtual machine's parent datastore.
Un-Tag Parent Host Complete	Check the condition after a company tag is removed from a virtual machine's parent host.
Un-Tag Parent Resource Pool Complete	Check the condition after a company tag is removed from a virtual machine's parent resource pool.
Un-Tag Request	Check the condition when an attempt is made to remove a company tag.
VDI Connecting to Session	Check the condition when a VDI session is started.
VDI Disconnected from Session	Check the condition when a VDI session is disconnected.
VDI Login Session	Check the condition when a user logs on to a VDI session.
VDI Logoff Session	Check the condition when a user logs off from a VDI session.
VM Analysis Complete	Check the condition when a SmartState Analysis of virtual machine completes.
VM Analysis Failure	Check the condition when a SmartState Analysis of virtual machine fails.

Event	Description
VM Analysis Request	Check the condition when a SmartState Analysis is requested from the Infrastructure Automation console.
VM Analysis Start	Check the condition when a SmartState Analysis of virtual machine is started.
VM C & U Processing Complete	Check the condition when the processing of capacity and utilization data has finished.
VM Clone Complete	Check the condition when a virtual machine is cloned.
VM Clone Start	Check the condition when a virtual machine clone is started.
VM Compliance Check	Check the condition when a compliance check is performed on a virtual machine.
VM Compliance Failed	Check the condition when a virtual machine fails a compliance check.
VM Compliance Passed	Check the condition when a virtual machine passes a compliance check.
VM Create Complete	Check the condition when a virtual machine is created.
VM Delete (from Disk)	Check the condition when a disk on a virtual machine is deleted.
VM Delete (from Disk) Request	Check the condition when someone tries to delete a virtual machine from disk from the user interface.
VM Guest Reboot	Check the condition when a virtual machine is rebooted.
VM Guest Reboot Request	Check the condition when someone tries to reboot a virtual machine from the Infrastructure Automation console.
VM Guest Shutdown	Check the condition when the operating system of a virtual machine shuts down.
VM Guest Shutdown Request	Check the condition when someone tries to shut down the operating system of a virtual machine from the user interface.
VM Live Migration (VMOTION)	Check the condition when a vMotion migration is performed.
VM Pause	Check the condition when a virtual machine is paused.
VM Pause Request	Check the condition when someone tries to pause a virtual machine from the Infrastructure Automation console.
VM Power Off	Check the condition when a virtual machine is turned off.
VM Power Off Request	Check the condition when someone tries to power off a virtual machine from the Infrastructure Automation console.
VM Power On	Check the condition when a virtual machine is turned on.
VM Power On Request	Check the condition when someone tries to turn on a virtual machine from the Infrastructure Automation console.
VM Provision Complete	Check the condition when a virtual machine is provisioned.
VM Remote Console Connected	Check the condition when a virtual machine is connected to a remote console.
VM Removal from Inventory	Check the condition when a virtual machine is unregistered.
VM Removal from Inventory Request	Check the condition when a request is sent from the Infrastructure Automation console to unregister a virtual machine.
VM Renamed Event	Check the condition when a virtual machine is renamed on its provider.
VM Reset	Check the condition when a virtual machine is restarted.
VM Reset Request	Check the condition when a virtual machine is restarted from the Infrastructure Automation console.
VM Resume	Check the condition when a virtual machine is resumed.
VM Retire Request	Check the condition when a virtual machine retirement request is created from Infrastructure Automation.
VM Retired	Check the condition when a virtual machine is retired.
VM Retirement Warning	Check the condition when a warning threshold is reached for retirement.
VM Settings Change	Check the condition when the settings of virtual machine are changed.
VM Shelve	Check the condition when a virtual machine is shelved.
VM Shelve Offload	Check the condition when a virtual machine is removed and deleted with the shelve offload operation.
VM Shelve Offload Request	Check the condition when a shelve offload request is created from Infrastructure Automation for a virtual machine.
VM Shelve Request	Check the condition when a virtual machine shelve request is created from Infrastructure Automation.

Event	Description
VM Snapshot Create Complete	Check the condition when a snapshot is completed.
VM Snapshot Create Request	Check the condition when someone tries to create a snapshot of a virtual machine from the user interface.
VM Snapshot Create Started	Check the condition when a snapshot creation is started.
VM Standby of Guest	Check the condition when the operating system of a virtual machine goes to standby.
VM Standby of Guest Request	Check the condition when someone tries to put the operating system of a virtual machine in standby from the Infrastructure Automation console.
VM Suspend	Check the condition when a virtual machine is suspended.
VM Suspend Request	Check the condition when someone tries to suspend a virtual machine from the Infrastructure Automation console.
VM Template Create Complete	Check the condition when a virtual machine template is created.

Events and Descriptions

Infrastructure Management - Administration

- Assigning a Custom Analysis Profile to a Virtual Machine
- General Configuration
- Appliance Hardening Guide
- Scripting Actions in Infrastructure Management
- Monitoring, Alerts, and Reporting
- Self Service User Interface Guide

- Monitoring, Alerts, and Reporting
  - Overview Dashboard
    - Adding a Widget
    - Resetting to the Default Set of Widgets
    - Removing a Widget
    - Zooming in to a Chart Widget
    - Opening a Chart or Report Widget in Full Screen View
    - Minimizing or Maximizing a Widget
    - Downloading a Report Widget as a File
  - Creating Dashboard Widgets
    - Creating a Report Widget
    - Creating a Chart Widget
    - Creating a Menu Widget
    - Editing a Widget
    - Copying a Widget
    - Deleting a Widget
    - Importing a Widget
    - Exporting a Widget
    - Generating Widget Content Immediately
  - Reports
    - Running Reports
      - Generating a Single Report
    - Scheduling a Report
      - Modifying a Report Schedule
      - Running a Scheduled Report Immediately
    - Viewing Reports
      - Changing Report Views
        - Filtering Report Data
        - Sorting Reports by Column
      - Downloading a Report
      - Showing a Report in Full Screen
    - Adding a Report
    - Copying a Report
    - Editing a Report

        - Deleting a Report
        - Importing a Report
        - Exporting a Report
        - Managing Report Menus
            - Managing Report Menu Accordions
            - Managing Report Menu Folders
            - Organizing Reports in Report Menus
    - Chargeback
        - Chargeback Rates
            - Memory Used Cost
            - CPU Total Cost
            - CPU Used Cost
            - Storage Allocated Cost
            - Storage Total Cost
            - Storage Used Cost
        - Creating Chargeback Rates
            - Editing a Chargeback Rate
            - Copying a Chargeback Rate
            - Adding a Chargeback Rate
        - Assigning Chargeback Rates
            - Configuring Chargeback for Compute
            - Configuring Chargeback for Storage
        - Creating a Chargeback Report
    - Optimization
        - Running an Optimization Report
        - Viewing Optimization Reports
    - Alerts
        - Assigning the Notifier Role
        - Creating an Alert
        - Creating a Hardware Reconfigured Alert
        - Creating a Normal Operating Range Alert
        - Creating a Real Time Performance Alert
        - Creating an Hourly Performance Alert
        - Creating a hostd Log Threshold Alert
        - Creating a VMware Alarm Alert
        - Creating an Expression Alert
        - Create an Operational Alert
        - Operational Alert Types
        - Editing an Alert
        - Copying an Alert
        - Deleting an Alert
        - Evaluating an Alert
    - Alert Profiles
        - Creating Alert Profiles
        - Editing an Alert Profile
        - Deleting an Alert Profile
        - Assigning an Alert Profile
    - Importing and Exporting
        - Importing and Exporting Policies, Policy Profiles, and Alerts
        - Exporting a Policy, Policy Profile, or an Alert
    - OpenShift Prometheus Alerts
        - Configuring Prometheus Alerts in Infrastructure Automation
            - Deploying and Configuring Prometheus on OpenShift
            - Assigning Prometheus Alert Profiles to the Enterprise
            - Adding an OpenShift Containers Provider with a Prometheus Alert Endpoint
- Regular Expressions
- Reportable Fields in Infrastructure Automation

Monitoring, Alerts, and Reporting

Overview Dashboard

Overview shows your virtual environment's events, reports, and configurable alerts. This information supports better information technology decision making and predictions for future virtual machine management.

When you log in to the console, it brings you directly to the **Overview Dashboard** page by default. The console uses widgets to organize this page, providing you with a default set of commonly used widgets. The configuration items on these widgets are clickable, allowing you to drill down directly to a referenced item.

Customize this page to include charts and reports you specifically want to see as soon as you log in to the console. You can add, remove, move, minimize, zoom into, and maximize widgets. Only users with the proper access can create widgets.

Adding a Widget

You can add widgets to the dashboard to accommodate the information you want to see upon login.

1. Navigate to **Overview > Dashboard**.
2. Click **(Add a Widget)**.
3. Select the widget you want to add from the list.

Note:

Only widgets that are not currently showing on the dashboard will appear in this list.

Resetting to the Default Set of Widgets

You can reset to default set of widgets according to your needs with the following steps:

1. Navigate to **Overview > Dashboard**.
2. Click **(Reset Dashboard Widgets to the defaults)**.

Removing a Widget

You can remove a widget when you no longer need the widget to be displayed in the dashboard.

1. Navigate to **Overview > Dashboard**.
2. From the widget that you want to remove, click **(Remove from Dashboard)** in the upper right corner of the widget.
3. Click **OK**.

The widget is removed from the dashboard, but it is not deleted. It can be added again if needed.

Zooming in to a Chart Widget

1. Navigate to **Overview > Dashboard**.
2. From the chart widget that you want to enlarge, click **(Zoom in)** in the upper right corner of the widget.

The enlarged chart is opened in its own pop-up window. To close this window, click **(Close)** in the upper right corner.

Opening a Chart or Report Widget in Full Screen View

1. Navigate to **Overview > Dashboard**.
2. From the chart or report widget that you want to enlarge, click **(Full Screen view)** in the upper right corner of the widget.
3. Click **OK**.

Minimizing or Maximizing a Widget

1. Navigate to **Overview > Dashboard**.
2. From the chart or report widget that you want to maximize, click **(Minimize)** or **(Full Screen)** in the upper right corner of the widget.

Downloading a Report Widget as a File

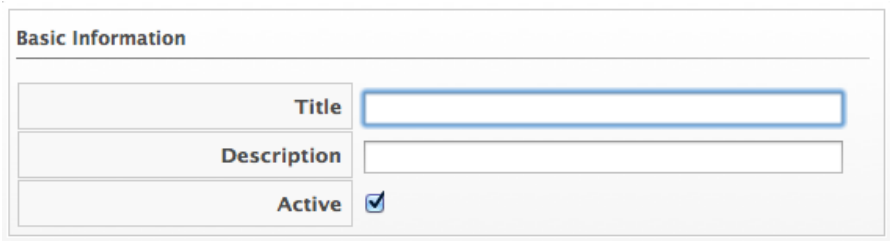
1. Navigate to **Overview > Dashboard**.

2. From the chart or report widget that you want to download as a PDF, click **[(Download PDF)** in the upper right corner of the widget.
3. Click **OK**.

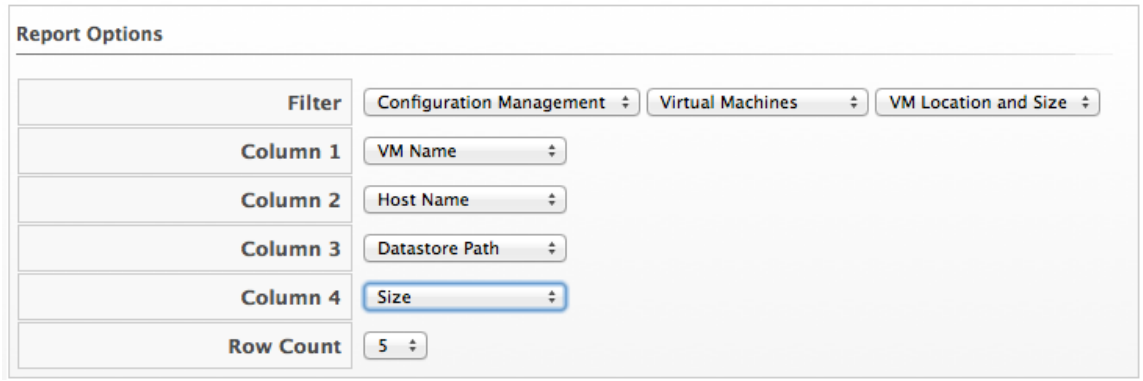
Creating Dashboard Widgets

Creating a Report Widget

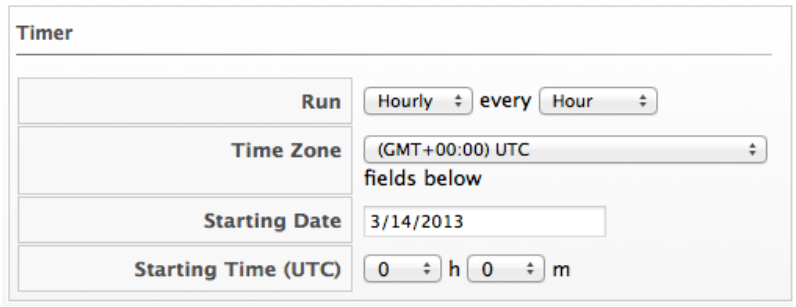
1. Navigate to **Overview > Reports**.
2. Click on the **Dashboard Widgets** accordion, then choose the **Reports** folder.
3. Click **(Configuration)**, then click **(Add a new Widget)**.
4. In the **Basic Information** area, type in a **Title** and **Description**. By default the widget will be active as soon as you create it. To make it inactive, uncheck the **Active** box.



5. From the **Report Options** area, select the filters until you are at the report you want for this widget. Select up to four columns from that report. Finally, for **Row Count**, select the number of rows that you want displayed.



6. In the **Timer** area, click the **Run** drop down to specify how often you want the widget data to get updated. The options displayed will depend on which **Run** option you choose. Select **Hourly**, **Daily**, **Weekly**, or **Monthly**.



7. Select a **Time Zone**. Type or select a date to begin the schedule in **Starting Date**. Select a **Starting Time** based on a 24 hour clock in the selected time zone.

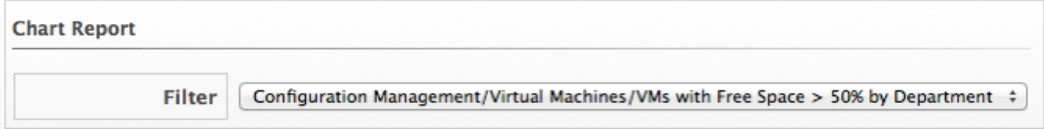
   Note:

   If you change the time zone, you will need to reset the starting date and time.

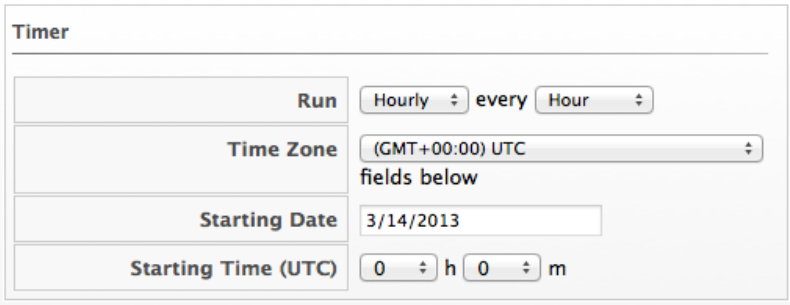
8. In the **Visibility** area, select **<To All Users>**, so that all users can use this widget no matter what user role they are assigned. Select **<By Role>** to assign this widget to specific user roles. Select **<By Group>** to assign this widget to specific groups.
9. Click **Add**.

Creating a Chart Widget

1. Navigate to **Overview > Reports**.
2. Click the **Dashboard Widgets** accordion and click the **Charts** folder.
3. Click **(Configuration)**, then click **(Add a new Widget)**.
4. In the **Basic Information** area, type in a **Title** and **Description**. By default the widget will be active as soon as you create it. To make it inactive, uncheck the **Active** box.
5. From the **Chart Report** area, select a chart to display in the widget.



6. In the **Timer** area, click the **Run** drop down to specify how often you want the widget data to get updated. The options displayed will depend on which **Run** option you choose. Select **Hourly**, **Daily**, **Weekly**, or **Monthly**.



7. Select a **Time Zone**. Type or select a date to begin the schedule in **Starting Date**. Select a **Starting Time** based on a 24 hour clock in the selected time zone.

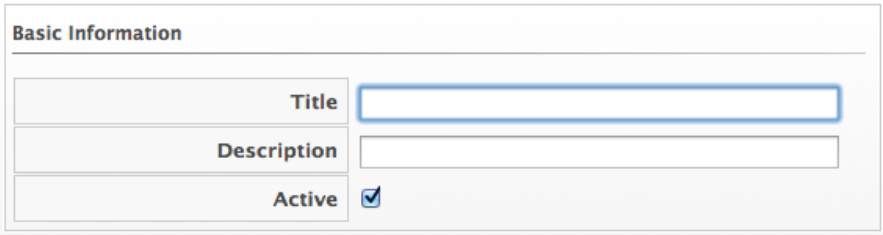
   Note:

   If you change the time zone, you will need to reset the starting date and time.

8. In the **Visibility** area, select **<To All Users>**, so that all users can use this widget no matter what user role they are assigned. Select **<By Role>** to assign this widget to specific user roles. Select **<By Group>** to assign this widget to specific groups.
9. Click **Add**.

Creating a Menu Widget

1. Navigate to **Overview > Reports**.
2. Click the **Dashboard Widgets** accordion and click the **Menus** folder.
3. Click **(Configuration)**, then click **(Add a new Widget)**.
4. In the **Basic Information** area, type in a **Title** and **Description**. By default the widget will be active as soon as you create it. To make it inactive, uncheck the **Active** box.



5. In the **Menu Shortcuts** area, use the **Add a Shortcut** dropdown to select all the places in the console that you want to add to this widget.
6. In the **Visibility** area, select **<To All Users>**, so that all users can use this widget no matter what user role they are assigned. Select **<By Role>** to assign this widget to specific user roles. Select **<By Group>** to assign this widget to specific groups.
7. Click **Add**.

Editing a Widget

1. Navigate to **Overview > Reports**.
2. Click the **Dashboard Widgets** accordion and select the widget you want to edit.
3. Click **(Configuration)**, and then **(Edit this Widget)**.
4. Make the required changes.
5. Click **Save**.

Copying a Widget

1. Navigate to **Overview > Reports**.
2. Click the **Dashboard Widget** accordion, and select the widget you want to copy.
3. Click **(Configuration)**, and then **(Copy this Widget)**.
4. Type a unique name for the widget and edit its properties.
5. Click **Save**.

Deleting a Widget

Note:

Default widgets cannot be deleted, but they can be copied.

1. Navigate to **Overview > Reports**.
2. Click the **Dashboard Widget** accordion and select the widget you want to delete.
3. Click **(Configuration)**, and then **(Delete this Widget from the Database)**.
4. Click **OK**.

Importing a Widget

You can share widgets between appliances using the export and import features.

1. Navigate to **Overview > Reports**.
2. In the **Import/Export** accordion, click **Widgets**.
3. In the **Import** area, click **Browse** to select an import file.
4. Click **Upload**.

Exporting a Widget

You can share widgets between appliances using the export and import features.

1. Navigate to **Overview > Reports**.
2. In the **Import/Export** accordion, click **Widgets**.
3. In the **Export** area, select the widgets that you want to export.
4. Click **Export**.

Generating Widget Content Immediately

1. Navigate to **Overview > Reports**.
2. Click the **Dashboard Widgets** accordion and select the widget you want to generate.
3. Click **(Configuration)**, and then **(Generate Widget Content now)**.
4. Click **OK**.

The content is generated immediately instead of waiting for the next scheduled update. Generation of widget content is shown under **Tasks**, accessed from the settings menu.

Reports

Click the **Reports** accordion under **Overview > Reports** to see a list of reports available. These reports have been constructed to help you view the most commonly requested and significant data. From here, you can also create reports if you have appropriate access. Infrastructure Automation provides a large group of default reports organized into categories. Each category has its own set of subfolders.

- **Configuration Management**: hardware, application, network, service, user account, operating system, and snapshot information for all of your items.
- **Migration Readiness**: information related to items required to migrate a virtual machine.
- **Operations**: free space on registered and unregistered virtual machines, power states for virtual machines, and SmartState analysis status.

  This category also provides reports relating to the operation of Infrastructure Automation, such as user IDs and snapshots taken by Infrastructure Automation.
- **VM Sprawl**: usage information and disk waste.
- **Relationships**: virtual machine, folder, and cluster relationships.
- **Events**: operations and configuration management events.
- **Performance by Asset Type**: performance of your virtual infrastructure.

  You must be capturing capacity and utilization data to get this information.
- **Running Processes**: information on processes running on a virtual machine.

  You must have domain credentials entered for the zone to collect the information for these reports, and the virtual machine must have been analyzed at least once.
- **Trending**: projections of datastore capacity, along with host CPU and memory use.
- **Tenants**: quotas report aggregated by each tenant that shows quota name, total quota, in use, allocated, and available. The report currently lists all tenants and there is no nesting information available by parent and child tenants.
- **Provisioning**: provisioning activity based on the approver, datastore, requester, and virtual machine.

For a complete list of reportable fields in Infrastructure Automation, see Reportable Fields.

Running Reports

There are two different ways to generate a report: by scheduling the report, or running the report manually by clicking the report generation button on the **Reports** page. Infrastructure Automation uses interactive report generation so that reports are placed on a queue. A visual indicator of the reports status is shown. All reports are automatically saved so that they can be downloaded and analyzed later.

Generating a Single Report

1. Navigate to **Overview > Reports**.
2. Click the **Reports > All Reports** accordion. Navigate to the report you want to view by selecting the appropriate folder.
3. Click (**Queue**).
4. The report generation is placed on the queue and its status shows in the reports page.

	Queued At	Run At	Source	User ID	Status
	10/21/11 14:14:11 UTC		Requested by user	admin	Queued
	10/21/11 14:00:08 UTC	10/21/11 14:00:22 UTC	Requested by user	admin	Finished

5. Click **(Reload current display)** to update the status.
6. When a report has finished generating, click on its row to view it.

Scheduling a Report

You can view historical data by creating reports on a scheduled basis. In addition, scheduled reports can be emailed directly to users.

Note:

You may need to disable, change the report filter, or change the frequency of a schedule. To do this, you will need to edit the schedule.

There are two ways to schedule a report:

Add a new Schedule through the **Reports** accordion:

1. Navigate to **Overview > Reports**.
2. Select a report from the **Reports** accordion and click **Configuration**.
3. Click **Add a New Schedule** or click the Schedules accordion and click **Configuration** > **Add a New Schedule**.

or

Add a new Schedule through the **Schedules** accordion:

1. Navigate to **Overview > Reports**.
2. Click the **Schedules** accordion.
3. Click **Configuration**.
4. Click **Add a New Schedule**.

Adding new Schedule details.

1. In the **Basic Information** area, type in a **Name** and **Description** for the schedule.
   - By default, **Active** is checked to enable the scan.
2. The **Report Selection** area is pre-populated if you added the schedule directly from the report. If you are adding from the schedule according, use the **Filter** drop downs to select the report that you want to schedule.
3. In the **Timer** area, click the **Run** drop down to specify how often you want the analysis to run. Your options after that will depend on which run option you choose.
   - Click **Once** to have the analysis run just one time.
   - Click **Daily** to run the analysis on a daily basis. You will be prompted to select how many days you want between each analysis.
   - Click **Hourly** to run the analysis hourly. You will be prompted to select how many hours you want between each analysis.
   - Type or select a date to begin the schedule in **Starting Date**.
   - Select a **Starting Time** based on a 24 hour clock in the Infrastructure Automation appliance's Time Zone.
4. Use the **E-Mail after Running** feature to send an email after the report has been generated to selected users. The email will include a link to the report.
   - Check **Send an E-mail**.
   - In **From (leave blank for default)**, type in the sending email.
   - Use **Add a User**, to select a specific user. The user must have a valid email address entered under accounts.
   - Use **Add (enter manually)** to type in the address not registered to a User.
   - Then, click **(Add)**.
5. If you are sending an email after the report runs, then you can select further options under Email Options.
   - Check **Send if Report is Empty** if you want an email even if no records exist in the report.
   - Next to Attachments, check if you would like the report attached as a **Text**, **CSV**, or **PDF** file.
6. Click **Add**.

Note:

See Outgoing SMTP Email Settings in the *General Configuration* guide, to learn how to verify the address and validate outgoing email settings.

Modifying a Report Schedule

1. Navigate to **Overview > Reports**
2. Click the **Schedules** accordion and select the schedule you want to edit.

3. Click (**Configuration**), then click (**Edit this Schedule**).
4. Make the required changes.
5. Click **Save**.

Running a Scheduled Report Immediately

1. Navigate to **Overview > Reports**.
2. Click the **Schedules** accordion and select the schedule you want to run.
3. Click (**Configuration**), then click (**Queue up this Schedule to run now**).

Viewing Reports

Once you have run a report, you can view it at any time after the first scheduled time has occurred.

1. Navigate to **Overview > Reports**.
2. Click the **Saved Reports** accordion or the **Reports** accordion.
3. Click on the instance of the report you want to view.

Changing Report Views

After running a report, you can customize your view in several ways.

Some reports can be viewed as charts as well as lists. This will depend on the type of data and on how the report has been created. Where applicable, you will see these additional buttons:

1. Navigate to **Overview > Reports**.
2. Click a completed report to view. Click one of the following buttons for the view you want:
   - Click for **Graph View**.
   - Click for **Hybrid View**.
   - Click for **Tabular View**.

Filtering Report Data

To filter the report information you see, use the **Data View** button.

1. Navigate to **Overview > Reports**.
2. Click a completed report to view.
3. Click the button for **Data View**, which opens the search field.
4. Select the criteria from the dropdown menu to filter by, and enter your search term.
5. Press **Enter** to search.

Sorting Reports by Column

In tabular and hybrid view, you can sort a list of reports by column.

1. Navigate to **Overview > Reports > Saved Reports**.
2. Click a report name to show a list of completed reports.
3. Click any blue column heading to sort the reports by the values in that column.

Downloading a Report

Download reports to analyze the data using other tools or to print the report.

When you click on one of the supplied reports, you are presented with a group of buttons to download the report in one of three formats or to view the report in a full screen.

1. Navigate to **Overview > Reports**.
2. Click the report you want to view.
3. Click on the row for the instance of the report you want to download. If the report needs to be generated, see Running Reports.
4. Click on the report download buttons for the type of export you want.
    - Click **(Download this report in text format)** to download as text.
    - Click **(Download this report in csv format)** to download as a comma-separated file.
    - Click **(Download this report in PDF format)** to download as PDF.
    - The report is automatically named with the type of report and date.

Note:

Edit and delete buttons are only visible to administrators and super administrators. Edit and delete functions are only available to customer-created reports. The Infrastructure Automation pre-configured reports cannot be edited or deleted, but they can be copied.

Showing a Report in Full Screen

View the report in full screen to examine report details. From full screen, you can also print the chart that accompanies a report.

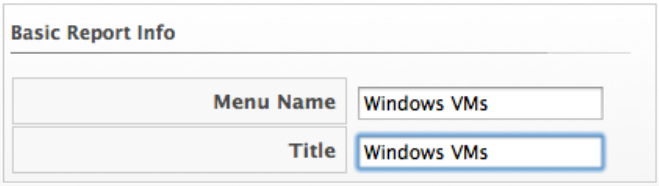
1. Navigate to **Overview > Reports**.
2. Click the **Reports > All Reports** accordion and select the report that you want to view.
3. Click the **Saved Reports** tab and select a report.
4. On the **Saved Report** detail screen, click (**Configuration**), and click (**Show full screen Report**).

The report will appear in full screen view. Click back on your browser to return to the reports explorer.

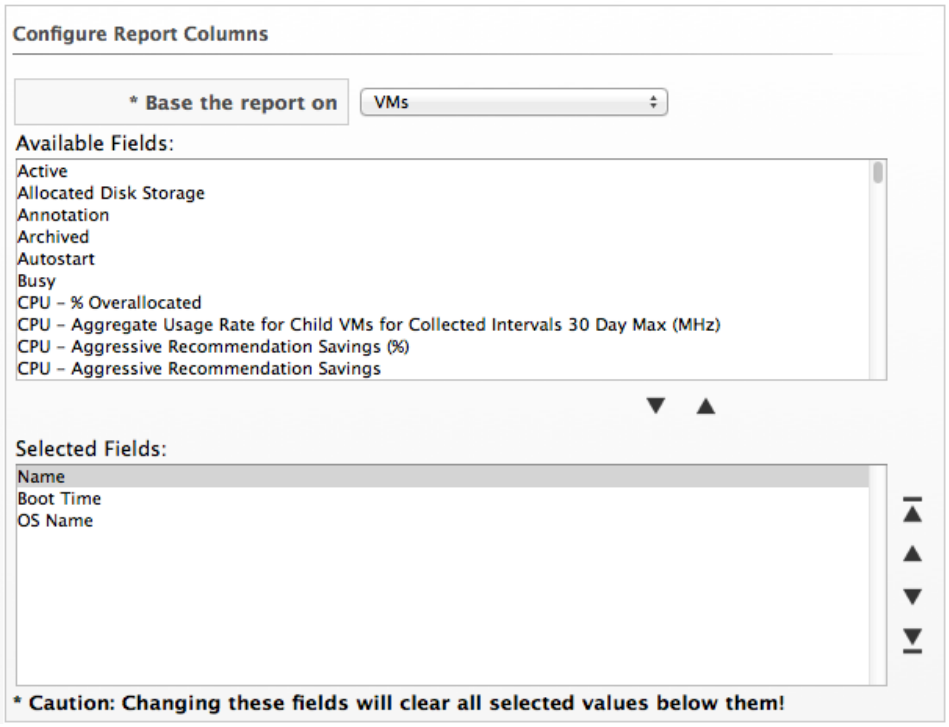
Adding a Report

Add reports if the default reports do not include what you need or you want to narrow the scope of a report. For example, you may want a report that shows only Windows virtual machines.

1. Navigate to **Overview > Reports**.
2. Click the **Reports > All Reports** accordion.
3. Click **(Configuration)**, then click **(Add a New Report)**.
4. In the **Columns** tab, edit the **Basic Report Info** area.



    - Type a unique name in **Menu Name** for how you want the report described in the menu list.
    - Type the **Title** you want displayed on the report in title.
5. Add fields in the **Configure Report Columns** area.



- Use the **Base the report on** table dropdown to choose a table to get fields from.

  Note:

  If you change the report base or the interval, all selections below will be reset.

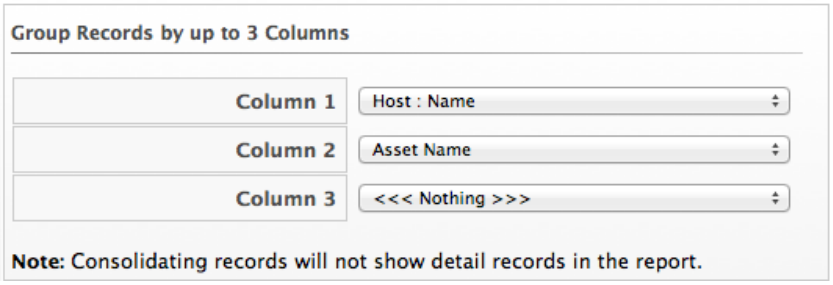
- Select fields that you want in the report from **Available Fields**, and then click **(Move selected fields down)**. In addition to the fields, you can also select any tags that you have created and assigned.

- Change the order of the fields in the report by clicking **(Move selected fields up)** or **(Move selected fields down)**.

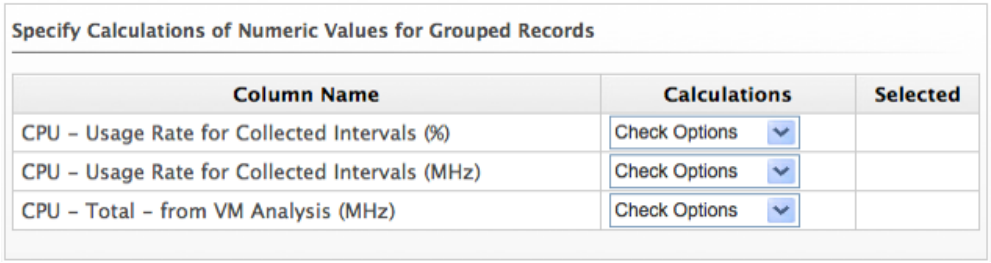
6. Click on the **Consolidation** tab to consolidate and aggregate data points into maximum, minimum, average, and total. Specifically, this would be useful for analyzing performance data over a specific period of time.

Note:

Please note that if you do this, you will not see individual records, but rather the calculation as a column header.



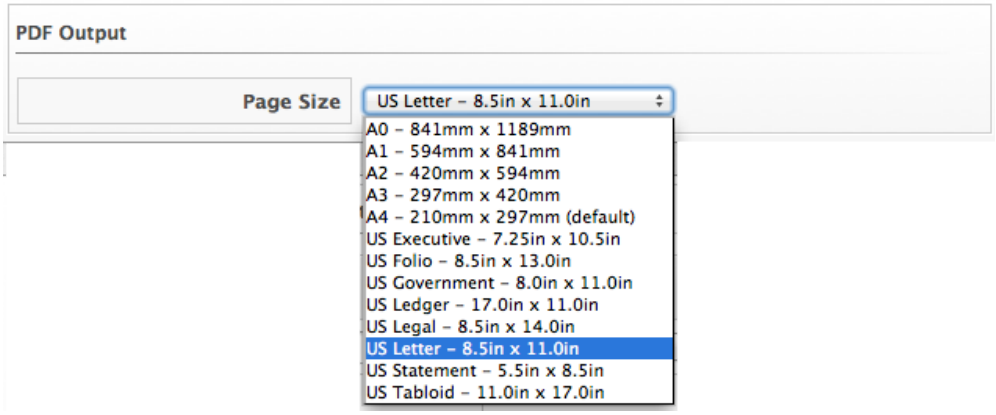
- Select the columns to group by.



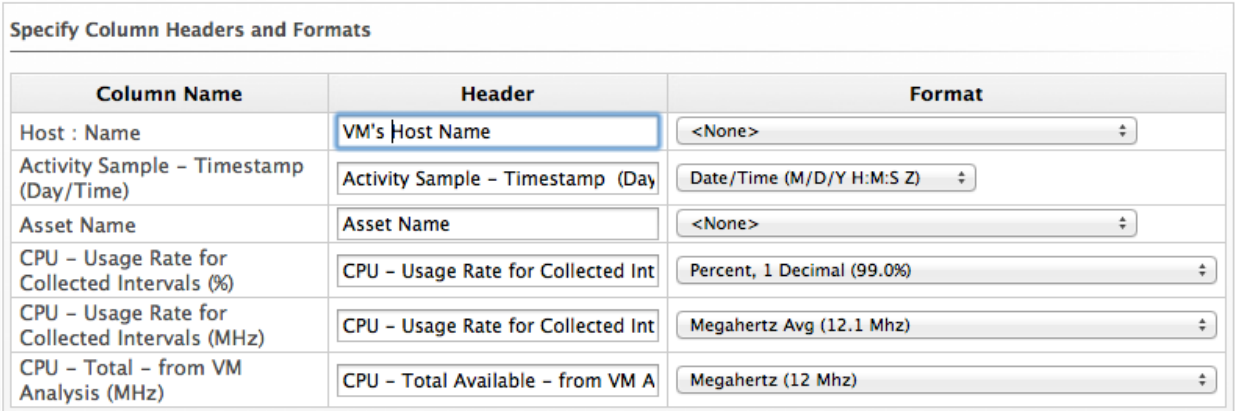
- For each numeric field selected in the report, you can click the dropdown under **Calculations**. Check the calculations you want to use.

7. Click on the **Formatting** tab to set the size of paper for a PDF and column header format.

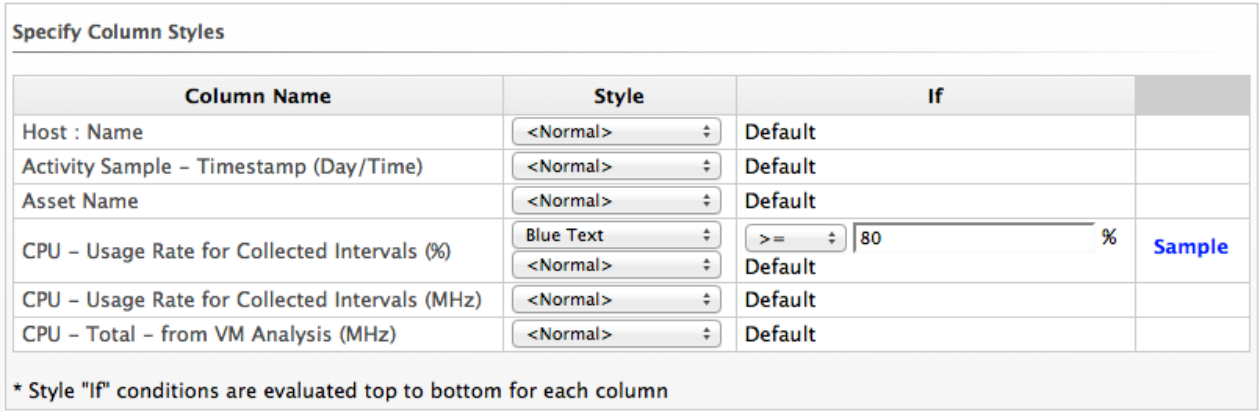
   - From the **PDF Output** area, select the page size from the dropdown.



   - From **Specify Column Headers and Formats**, type the text you want displayed for each field. For each numeric field, you can also set the numeric format.



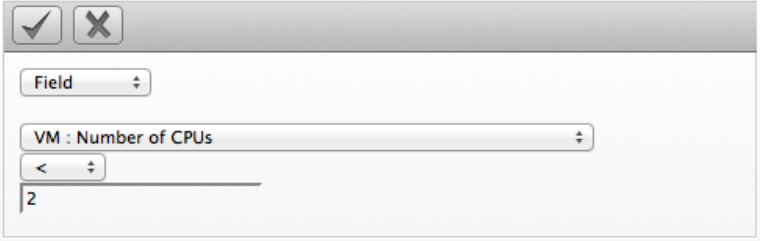
8. Click on the **Styling** tab to change the color of the text or the background for a row based on a condition.



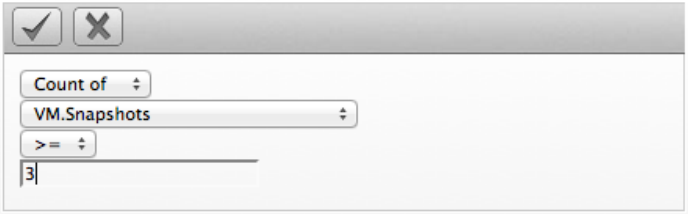
   - Use **Style** to select the format for the value. You can choose to change the text color or the background.
   - Use **If** to create a conditional statement for the style.

9. Click on the **Filter** tab to set filters for the data displayed in the report. There are two types of filters: the first is the **Record Filter** which is the primary filter of the main tables records, the second is also a **Display Filter**, which is a secondary filter of rows based on the fields of the child table. Click in the appropriate area to use the expression editor to choose the appropriate options for your criteria. Based on what you choose, different options will show.

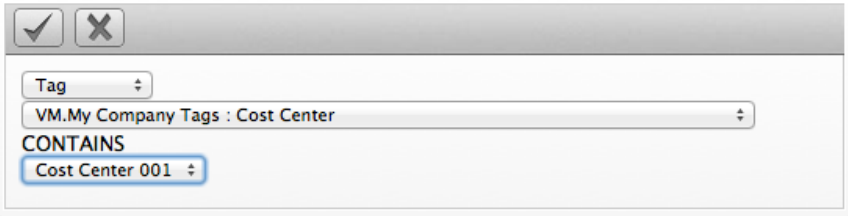
   - Click **Field** to create criteria based on field values.



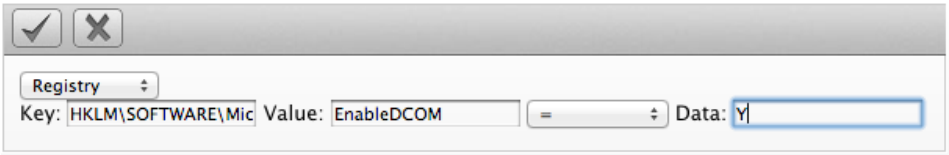
- Click **Count of** to create criteria based on the count of something, such as the number of snapshots for a virtual machine, or the number of virtual machines on a host.



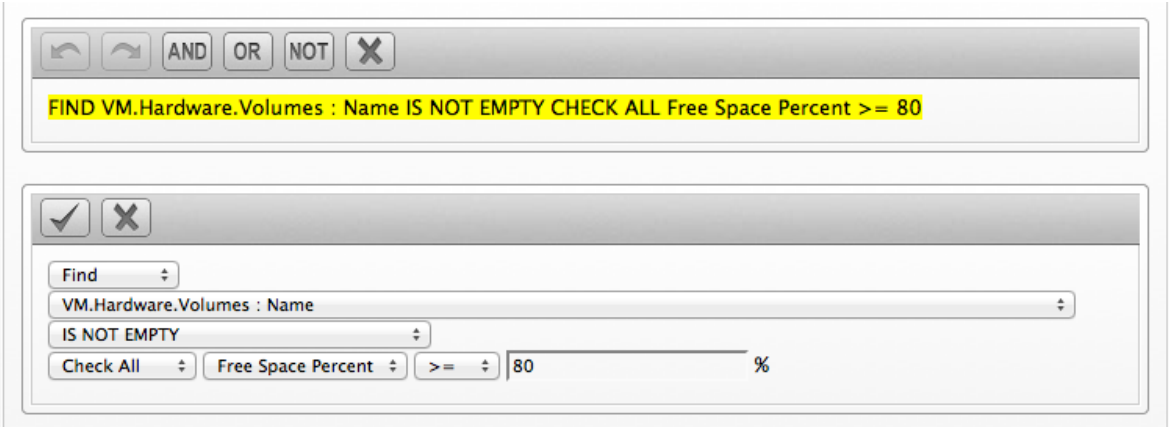
- Click **Tag** to create criteria based on tags assigned to your resources. For example, you may want to check the power state of a virtual machine or see if it is tagged as production.



- Click **Registry** to create criteria based on registry values. For example, you may want to check if DCOM is enabled on a Windows system. (Note that this applies only to Windows operating systems.)



- Click **Find** to seek a particular value, and then check a property. For example, finding the **`Admin`** account and checking that it is enabled.

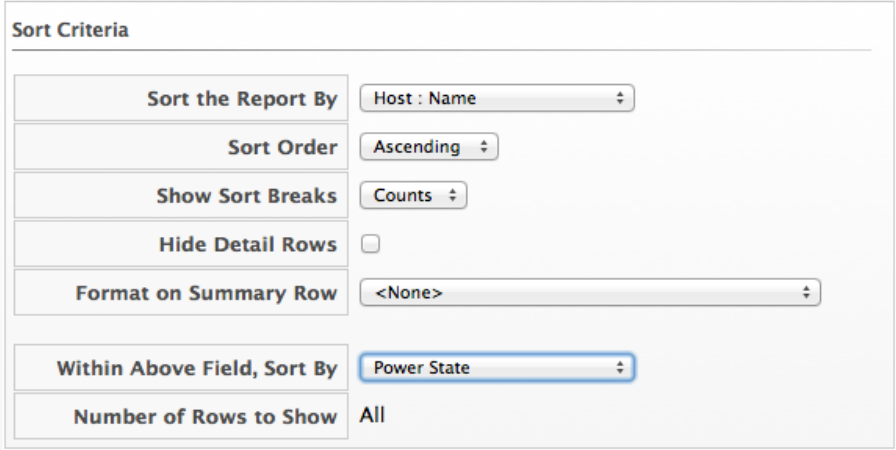


- Click ✓ **`(Commit Expression Element Changes)`** to add the expression.

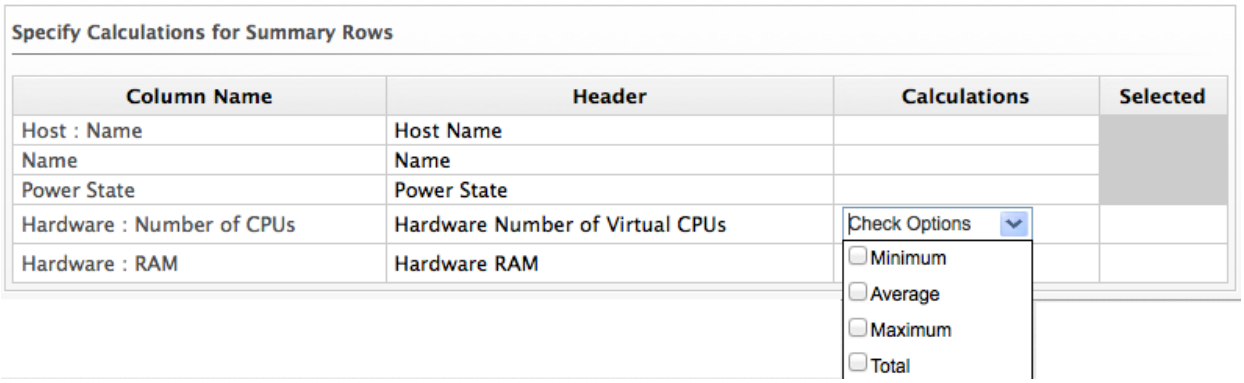
  Note:

  The filters that you apply will show at the bottom of the report so that you know which filters have been applied.

10. Click on the **Summary** tab to select sort order, sort type, groupings, and group calculations for the report. **Summary** groups rows of data based on the sort breaks. You can only sort by fields that display in the report.

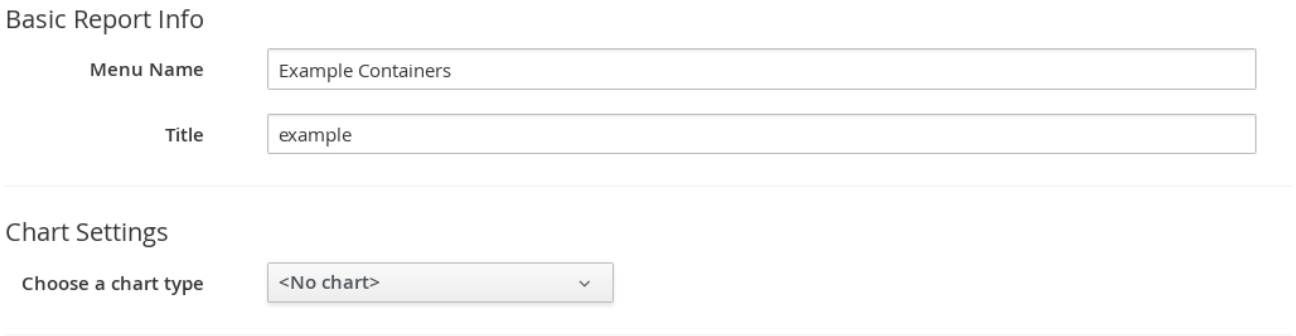


- Set the primary sort in **Sort the Report by**.
- Set the next sorts in **Within Above Field, Sort By**.
- Select the type of sort, ascending or descending, in **Sort order**.
- In **Show Sort breaks**, select **Yes** to show the sort breaks, **Counts** to show sort breaks with the count, or **No** for no sort breaks.



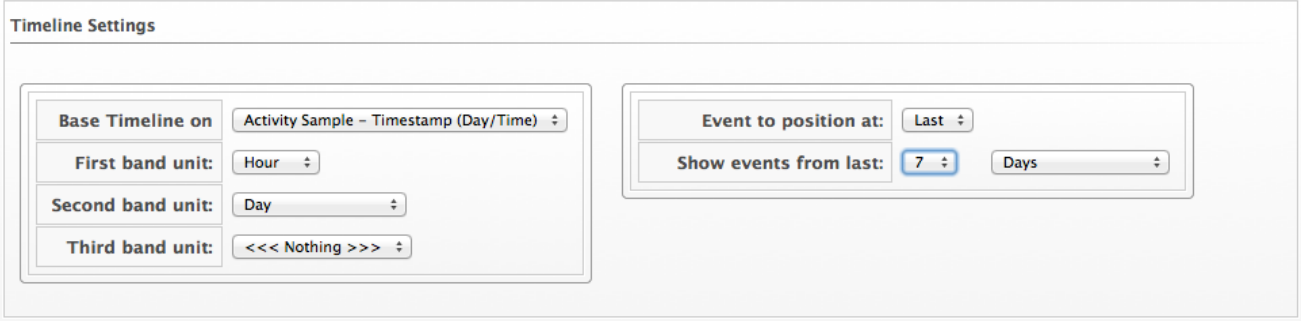
- For any numeric field, you can select to show minimum, average, maximum, and total in the sort break.

11. Click on the **Charts** tab to create a chart for the report. This is not required.



- Use **Choose a chart type** to select a type of chart. Note that some charts may not produce the result you are looking for based on the types of fields in the report and its sort order.
- If you only want to see the top values, select the number of top values from **Top values to show**.
- If you want to see the total number of values that are not categorized or evaluated against others, check **Sum 'Other' values**.

12. Click on the **Timeline** tab to select a timeline for the report. You must have a field of time or date format to use this feature.



- Use **Base Timeline on** to select a column in date or time format for the report.
- Select a unit of time for the first band in **First band unit**.
- Select a unit of time for the second band in **Second band unit**.
- Select a unit of time for the third band in **Third band unit**.
- Select an **Event to position at**.
- Select the range for the event to position from **Show events from last**.

13. Click the **Previews** tab to see a sample of your report.
14. When you have the report that you want, click **Add** to create the new report.

Note:

After the new report is created, to make the report accessible from the report menu, you must add it to a report menu.

Copying a Report

Use this feature to copy a report that is similar to one that you want to create. By copying a report, you only need to make minor edits instead of creating an entirely new report.

1. Navigate to **Overview > Reports**.
2. Click the **Reports > All Reports** accordion and select the report that you want to copy.
3. Click (**Configuration**), then click (**Copy this report**).
4. On the Columns tab, edit the **Basic Report Info** area to include a new **Menu Name**. Each **Menu Name** must be unique.
5. Make any other changes you need. See “Adding a Report” for details on the changes you can make.
6. Click **Add**.

Editing a Report

If you find that a report is not giving you the data that you need, you can edit it after it has been created. Note that only reports that you have created can be modified. Only administrators and super administrators of Infrastructure Automation can add, copy, edit, and delete reports.

1. Navigate to **Overview > Reports**.
2. Click the **Reports > All Reports** accordion and select the report you want to edit.
3. Click (**Configuration**), (**Edit this Report**).
4. Make any changes you need.
5. Click **Save**.

Deleting a Report

Delete reports when you find that they are no longer useful. Only administrators and super administrators of Infrastructure Automation can add, copy, edit, and delete reports.

Note:

Please note that only customer-created reports can be deleted.

1. Navigate to **Overview > Reports**.

2. Click the **Reports > All Reports** accordion and select the report you want to delete.
3. Click (**Configuration**), (**Delete this Report from the Database**).

   Note:

   The **Delete this report from the Database** option will only appear on reports you have created. Default reports cannot be deleted.
4. Click **OK**.

Importing a Report

Reports are stored in the VMDB; however, you can share customized reports among VMDBs. To do this use the export and import feature of reports.

1. Navigate to **Overview > Reports**.
2. Select **Custom Reports** from the **Import/Export** accordion.
3. In the **Import** area, click **Browse** to select an import file.
4. To overwrite an existing report with the same menu name, select **Overwrite existing reports**.
5. Click **Upload** to import the report to the **Company-Custom** folder.

Exporting a Report

Reports are stored in the VMDB. Custom reports may be shared among VMDBs. Share your custom reports using the export and import features.

1. Navigate to **Overview > Reports**.
2. Select **Custom Reports** from the **Import/Export** accordion.
3. In the **Export** area, select the reports to export.
4. Click **Export**.

Managing Report Menus

By default, all account roles have the same reports available. If you have the super administrator or administrator role, you can customize the accordions, the folders within them, and the locations of your reports.

Note:

When you create your own report, the report is not automatically available. You must add it to a report menu.

The report menu is a hierarchical structure that consists of the following components:

- The top level, under which the accordions are shown. (Top level is only displayed when you are in the report menu editor.)
- Accordions that are general categories for the reports. The defaults supplied are **Configuration Management**, **Migration Readiness**, **Operations**, **VM Sprawl**, **Relationships**, **Events**, **Performance by Asset Type**, **Trending**, **Tenants**, **Provisioning**, and **Running Processes**.
- Folders that are used to further organize reports within an accordion. For example, under the **Configuration Management**, there are folders for virtual machines, hosts, and other virtual infrastructure components.
- Reports that are stored directly in the folders.

Managing Report Menu Accordions

1. Navigate to **Overview > Reports**.
2. Click the **Edit Report Menus** folder.
3. Click the role whose menus you want to customize.
4. Click on **Top Level** to organize, add, and delete folders.
   - Click **(Move selected folder to top)** to move the folder to the top of the list.
   - Click **(Move selected folder up)** to move the folder up.

- Click **(Move selected folder down)** to move the folder down.
- Click **(Move selected folder to bottom)** to move the folder to the bottom of the list.
- Click **(Delete selected folder and its contents)** to delete a folder.
- Click **(Add folder to selected folder)** to add an folder.

  Note:

  If you are creating a new folder, **Top Level** must be selected under **Reports**. Be sure to select the folder you want to create a subfolder for on the left pane. To name the folder, double-click on **New Folder**, then click on **Top Level** in the **Reports** area.

5. When you are finished adding folders, click **Commit**. To revert, click **Discard**.
6. Click **Save**.

The new folder is added, and you can add folders in which to store reports to it. You can also organize the reports into folder that are meaningful to you.

Managing Report Menu Folders

1. Log in to the console as a user who is assigned either the super administrator or administrator account role.
2. Navigate to **Overview > Reports**.
3. Click the **Edit Report Menus** accordion.
4. Click the role whose menus you want to customize.
5. Click on the accordion name you want to organize or add folders to.
   - Click **(Move selected folder to top)** to move the folder to the top of the list.
   - Click **(Move selected folder up)** to move the folder up.
   - Click **(Move selected folder down)** to move the folder down.
   - Click **(Move selected folder to bottom)** to move the folder to the bottom of the list.
   - Click **(Delete selected folder and its contents)** to delete an folder.
   - Click **(Add subfolder to selected folder)** to add a folder. When creating a subfolder, be sure to select the folder that you want the subfolder to show under. To name the folder, double-click on **`New Folder`**.
6. When you are finished making changes click **Commit**. To revert, click **Discard**.
7. Click **Save**.

Note:

Only reports that are not already in another folder can be assigned.

Organizing Reports in Report Menus

1. Log in to the console as a user who is assigned either the Super Administrator or Administrator Account Role.
2. Navigate to **Overview > Reports**.
3. Click the **Edit Report Menus** accordion.
4. Click the role whose menus you want to customize.
5. Expand the **Report** accordion and menus using the triangles to the left of the item name until you are able to select the subfolder where you want to put reports.

6. Choose one of the following actions:
    - To add a report, select a report from the **Available Reports** area on the right and click **(Move selected reports left)**.
    - To remove a report from a folder, select the report from the **Selected Reports** area and click **(Move selected reports right)**.
    - To move a report to the top of the folder, select the report and click **(Move selected reports to top)**.
    - To move a report up one place in the folder, select the report and click **(Move selected reports up)**.
    - To move a report down one place in the folder, select the report and click **(Move selected reports down)**.
    - To move a report to the bottom of the folder, select the report and click **(Move selected reports to bottom)**.
7. When you are finished making changes click **Commit**.
8. Click **Save**.

The changes are made. The next time a user with this Account Role logs in, the new report and menu structure will be displayed.

Chargeback

The chargeback feature allows you to calculate monetary virtual machine charges based on owner or company tag. To use this feature you must be collecting capacity and utilization data.

Chargeback Rates

Infrastructure Automation provides a default set of rates for calculating chargeback costs, but you can create your own set of computing and storage costs by navigating to **Overview > Chargeback** and clicking the **Rates** accordion.

You can configure chargeback rates for various resources by selecting either **Compute** or **Storage** in the **Rates** accordion. **Compute** sets chargeback rates for CPU, disk I/O, fixed compute cost, memory, and network I/O items, while **Storage** sets chargeback rates for fixed storage cost and disk storage.

Chargeback costs are computed using a set formula based on hourly cost per unit and hourly usage.

Chargeback can be calculated in the following currencies:

- Australian dollar (AUD)
- Brazilian real (BRL)
- Swiss franc (CHF)
- Chinese yuan renminbi (CNY)
- Euro (EUR)
- Pound sterling (GBP)
- Indian rupee (INR)
- Japanese yen (JPY)
- South Korean won (KRW)
- Mexican peso (MXN)
- Norwegian krone (NOK)
- Russian ruble (RUB)
- Swedish krona (SEK)
- Turkish lira (TRY)
- United States dollar (USD)
- South African rand (ZAR)

Chargeback rates can be assigned at a single rate or by tiers, where rates are assigned in ranges depending on level of usage.

Additionally, chargeback can be calculated at one fixed rate, or by a combination of fixed and variable rates per tier. Fixed rates are charged once per unit of time, and the variable rate is calculated by the level of usage multiplied by the number of resources used in a unit of time.

Memory Used Cost

Calculating the Memory Used Cost in dollars ($) for a day can be expressed in the following ways:

- Memory allocation per hour (in MB) * Hourly Allocation cost per megabyte * Number of Memory Allocation metrics available for the day
- Sum of Memory allocation for the day (in MB) * Hourly Allocation cost per megabyte
- Sum of Memory allocation for the day (in MB) * Daily Allocation cost per megabyte / 24

Memory costs can be measured in B, KB, MB, GB, or TB.

In a scenario where 9.29 GB of memory is used in a day with the chargeback rate set at one dollar ($1) per megabyte per day, the Memory Used Cost would be $396.42.

- 9.29 GB = 9514.08 MB
- 9514.08 MB * $1 (per MB per day) = $9514.08
- $9514.08 / 24 = $396.42 Memory Used Cost

CPU Total Cost

The CPU Total Cost is defined as the number of virtual CPUs over the selected interval (hour, day, week, month). CPU costs can be measured in units of Hz, KHz, MHz, GHz, or THz, as specified when creating a chargeback rate.

In a scenario where 16 CPUs are used in a day with the chargeback rate set at one dollar per CPU per day, the CPU Total Cost would be $16.

- 16 CPUs * $1 (per CPU per day) = $16 CPU Total Cost

CPU Used Cost

The CPU Used Cost is defined as the average CPU used in MHz over the selected rate interval (hour, day, week, month). CPU Used Cost is not supported for containers providers.

In a scenario where 2.5 GHz is used in a day with the chargeback rate set at $0.01 per MHz per day, the CPU Used Cost would be $25.

- 2.5 GHz = 2500 MHz
- 2500 MHz * $0.01 (per MHz per day) = $25 CPU Used Cost

Storage Allocated Cost

The Storage Allocated Cost is defined as the Allocated Disk Storage in bytes over the selected rate interval (hour, day, week, month). Storage costs can be measured in B, KB, MB, GB, or TB.

In a scenario where 500 GB are used in a day with the chargeback rate set at $0.10 per GB per day, the Storage Allocated Cost would be $50.

- 536,870,912,000 bytes = 500 GB
- 500 GB * $0.10 (per GB per day) = $50 Storage Allocated Cost

Storage Total Cost

The Storage Total Cost is defined as the Used Disk Storage in bytes over the selected rate interval (hour, day, week, month).

In a scenario where 250 GB are used in a day with the chargeback rate set at $0.10 per GB per day, the Storage Total Cost would be $25.

- 268,435,456,000 bytes = 250 GB
- 250 GB * $0.10 (per GB per day) = $25 Storage Total Cost

Storage Used Cost

The Storage Used Cost is defined as the Used Disk Storage in bytes over the selected rate interval (hour, day, week, month).

In a scenario where 250 GB are used in a day with the chargeback rate set at $0.10 per GB per day, the Storage Used Cost would be $25.

- 268,435,456,000 bytes = 250 GB

- 250 GB * $0.10 (per GB per day) = $25 Storage Used Cost

Note:

The following chargeback rates are not supported for containers providers:

- Allocated CPU count
- Used CPU
- Used disk I/O
- Allocated memory

Creating Chargeback Rates

Infrastructure Automation allows you to create your own set of computing and storage costs to use for billing.

Chargeback rates can be configured at a single rate or in tiers, where one rate is assigned to one usage range, and another rate is assigned to a different usage range. You can also assign fixed and variable rates per tier if desired.

There are three methods to create a chargeback rate: editing or copying an existing rate, or creating a new rate.

Editing a Chargeback Rate

1. Navigate to **Overview > Chargeback**.
2. Click the **Rates** accordion and select an existing rate from under **Compute** or **Storage**.
3. Click **(Configuration)** and **(Edit this Chargeback Rate)**.
4. Edit the rate as desired.
5. Click **Save** at the bottom of the screen to save the chargeback rate.

Copying a Chargeback Rate

1. Navigate to **Overview > Chargeback**.
2. Click the **Rates** accordion and select an existing rate from under **Compute** or **Storage**.
3. Click **(Configuration)** and **(Copy this Chargeback Rate)**.
4. Edit the rate as desired.
5. Click **Add** at the bottom of the screen to save the chargeback rate.

Adding a Chargeback Rate

The following example shows a scenario to create a new chargeback rate for allocated CPU usage in Euro per week measured by the following tiers, calculating both fixed and variable costs:

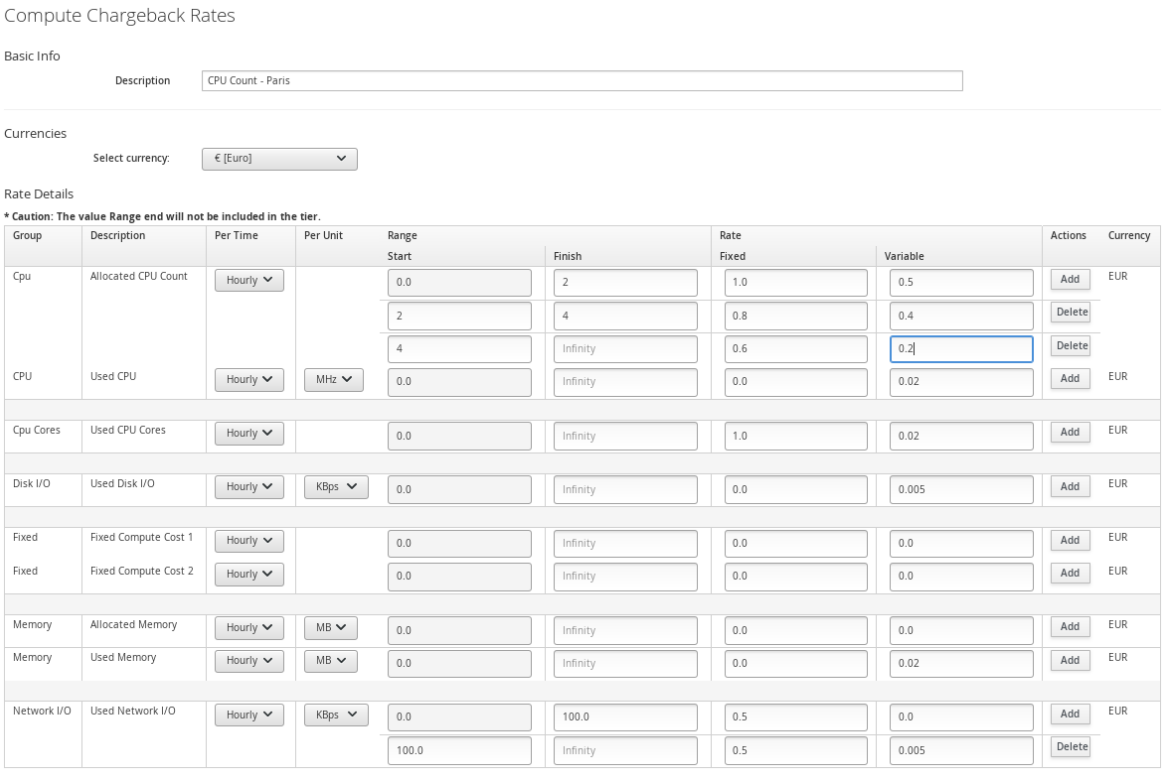
- Tier 1: 0-2 CPU = 1 Euro fixed + 0.5 Euro variable
- Tier 2: 2-4 CPU = 0.8 Euro fixed + 0.4 Euro variable
- Tier 3: 4 CPU and above = 0.6 Euro fixed + 0.2 Euro variable

To configure this rate:

1. Navigate to **Overview > Chargeback**.
2. Click the **Rates** accordion and select **Compute** to create a CPU chargeback rate.
    1. Click **(Configuration)** and **(Add a new Chargeback Rate)**.
3. Type in a **Description** for the chargeback rate.
4. Select **Euro** for currency.
5. In **Rate Details**, select the time, and fill in the range and rates for each tier. Alternatively, a single chargeback rate can be configured by using only one tier. For this example, in **Allocated CPU Count**, configure:
    1. For tier 1, set:

1. In **Per Time**, select **Hourly**.
2. In **Range**, specify the CPU range: **Start** remains as `0.0` and **Finish** is `2`.
3. In **Rate**, specify `1` for the **Fixed** rate, and `0.5` for the **Variable** rate.
4. Click **Add** under **Actions** to create a new, blank **Allocated CPU Count** line for tier 2.

2. For tier 2, set:
    1. In the new line in **Range**, specify the CPU range: **Start** is `2` and **Finish** is `4`.
    2. In **Rate**, specify `0.8` for the **Fixed** rate, and `0.4` for the **Variable** rate.
    3. Click **Add** under **Actions** to create another new, blank **Allocated CPU Count** line for tier 3.
3. For tier 3, set:
    1. In the new line in **Range**, specify the CPU range: **Start** is `4` and leave **Finish** as `Infinity`.
    2. In **Rate**, specify `0.6` for the **Fixed** rate, and `0.2` for **Variable**.



6. Click **Add** at the bottom of the screen to save the chargeback rate.

The rate can now be assigned.

Assigning Chargeback Rates

Infrastructure Automation allows you to assign chargeback rates by choosing from **Compute** and **Storage**. You can assign either a custom chargeback rate, or a default rate for Compute, Storage, or Container Images.

Note:

The **Default Container Image Rate** can be useful when applied to a containers provider with multiple images, where only some of the images have a chargeback rate assigned; the images without a custom rate will then use the default rate.

1. Navigate to **Overview > Chargeback**.
2. Click the **Assignments** accordion, and click either **Compute** or **Storage**.

Configuring Chargeback for Compute

Use **Compute** to assign a compute chargeback rate. You can assign chargeback rates to **The Enterprise**, **Selected Cloud/Infrastructure Providers**, **Selected Clusters/Deployment Roles**, **Selected Containers Providers**, **Tagged VMs and Instances**, **Tagged Images**, **Labeled**

Images, or **Tenants**.

1. From the **Basic Info** area, use the **Assign To** list to select a type of assignee to assign the rate set to.
2. For **Tagged VMs and Instances** and **Tagged Images** select a **Tag Category**.
3. For each item to set, select the chargeback rate to use. The options displayed vary based on the type you select.
4. Click **Save**.

Configuring Chargeback for Storage

Use **Storage** to assign a storage chargeback rate. You can assign chargeback rates to **The Enterprise**, **Selected Datastores**, **Tagged Datastores**, or **Tenants**.

1. From the **Basic Info** area, use the **Assign To** list to select a type of assignee to assign the rate set to.
2. For **Tagged Datastores** select a **Tag Category**.
3. For each item to set, select the chargeback rate to use. The options displayed vary based on the type you select.
4. Click **Save**.

The rate is assigned. The next time you generate a chargeback report, these values will be used.

Note:

When viewing chargeback, there is a rate for a virtual machine for the number of the CPUs. The chargeback for this parameter is calculated based on when the virtual machine is running. If the virtual machine is not running, then it is not charged for CPU allocation.

Creating a Chargeback Report

Infrastructure Automation allows you to create chargeback reports to monitor costs you charged.

Note:

When creating a chargeback report, the **Total Cost** is the complete cost for the chosen workload over the selected time frame. Total costs are calculated by adding the calculated costs for the amount used and the amount allocated for the time frame. When adding cost related columns to display on your report, be sure to select all rate assigned fields to present a comprehensive accounting of total costs.

1. Browse to menu: **Overview > Reports**.
2. Click the **Reports** accordion.
3. Click **Configuration**, then click **Add a new Report**.
4. On the **Columns** tab, fill out the **Basic Report Info** area.
   - Type a unique name in **Menu Name** for how you want the report described in the menu list.
   - Type the **Title** to display on the report.
5. Add fields in the **Configure Report Columns** area.
   - From the **Base the report on** list, select **Chargebacks**.
   - Select the fields to include in the report from the **Available Fields** list, then click **Move selected fields down**. In addition to the fields, you can also select any tags that you have created and assigned.
   - Change the order of the fields in the report by clicking **Move selected fields up** or **Move selected fields down**.
6. Click the **Formatting** tab to set the size of paper for a PDF and column header format.
   - From the **PDF Output** area, select the page size from the **Page Size** list.
   - From **Specify Column Headers and Formats**, type the text to display for each field. For each numeric field, you can also set the numeric format.
7. Click the **Filter** tab to set filters for the data displayed in the report.
   - From **Chargeback Filters**, select how you want the costs to show, the tag category, the tag, and how you want the items grouped.
   - From **Chargeback Interval**, select the time interval. You must have a full interval worth of data in order to select an option other than **Partial** in the **Daily Ending With** list.

8. Click the **Preview** tab, and then **Load** to see what the report will look like.
9. When you are satisfied that you have the report that you want, click **Add** to create the new report.

The new report is created. To make the report accessible from the **Report** menu, you must add it to a report menu.

Optimization

Optimization provides administrators with a series of reports to provide meaningful information on areas for optimizing resources such as CPU, memory and disk and identify potential savings.

Running an Optimization Report

1. Navigate to **Overview > Optimization**.
2. Click **Queue Report** for each report type you choose to run.

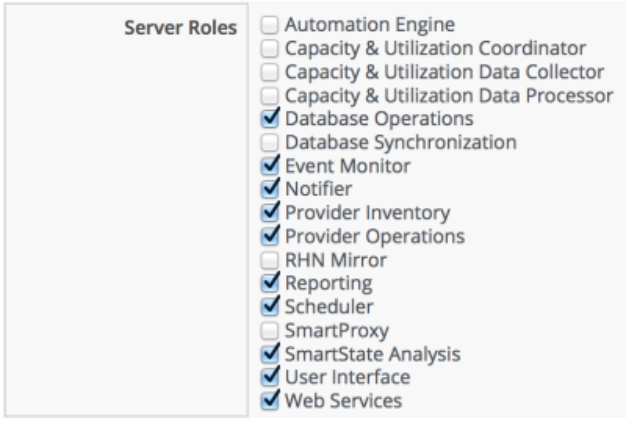
Viewing Optimization Reports

1. Navigate to **Overview > Optimization**.
2. Click on the report type you wish to view details for.
3. On the report type screen, click on a report instance to view details.
4. Sort reports by clicking on the field names at the top of the report.

Alerts

Assigning the Notifier Role

1. Click (**Configuration**).
2. Click the **Settings** accordion, and select the Infrastructure Automation server.
3. From the **Server Control** tab, select the **Notifier** role.



4. Click **Save**.

Creating an Alert

This section outlines the basic procedure to create an alert.

Note:

To send emails or SNMP traps from the Infrastructure Automation server, you must enable the Notifier server role and set up SMTP email or SNMP traps. For more information, see *General Configuration*.

1. Navigate to **Control > Explorer**.
2. Click the **Alerts** accordion, then click (**Configuration**), (**Add a New Alert**).
3. Enter the basic details of the alert:
    1. Enter a description in the **Description** field.

2. Select the **Active** check box to enable the alert after creation.
3. Select the severity level from the **Severity** list.
4. Select the inventory item on which to base the alert from the **Based On** list.
5. Select the type of event that triggers the alert from the **What to Evaluate** list.
6. Select the frequency with which to be notified if the event log threshold is reached from the **Notification Frequency** list.

4. Configure the parameters of the alert.

   Note:

   The available parameters depend on the options you selected in the **Based On** and **What to Evaluate** lists. See later sections for the details of these parameters.

5. Optionally, select **Send an E-mail** to configure options so that an email is sent when the alert is triggered:
   1. Enter the email address from which to send the email in the **From** field.
   2. Select a user from the **Add a User** list to add a user registered in Infrastructure Automation. The user must have a valid email address entered under accounts.
   3. Enter the email address of a user in the **Add (enter manually)** field and click + to add a user not registered in Infrastructure Automation.
6. Optionally, select **Send an SNMP Trap** to configure options so that an SNMP trap is sent when the alert is triggered:
   1. Enter the IP addresses of the hosts to send the trap in the **Host** fields.
   2. Select the version of SNMP to use from the **Version** list:
      1. If you select **v1**, enter a trap number in the **Trap Number** field. Enter **1**, **2**, or **3**, based on the appropriate suffix number in table_title.
      2. If you select **v2**, enter a trap object ID in the **Trap Object ID** field. Enter **info**, **warning**, or **critical** based on the values in table_title.

Object ID	Suffix Number Added to PEN	PEN with the Suffix Added
info	1	1.3.6.1.4.1.33482.1
warn, warning	2	1.3.6.1.4.1.33482.2
crit, critical, error	3	1.3.6.1.4.1.33482.3

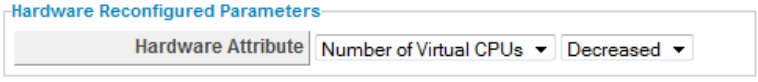
SNMP Trap Identifiers

7. Optionally, select **Send a Management Event** to trigger an automation event:
   1. Enter the name of the event that exists in the **Process/Event Class** in the **Event Name** field.
8. Click **Add**.

Creating a Hardware Reconfigured Alert

Use a hardware reconfigure alert to detect changes to the amount of memory or the number of CPUs on a virtual machine.

1. Navigate to **Control > Explorer**.
2. Click the **Alerts** accordion, then click (**Configuration**), (**Add a new Alert**).
3. In the **Info** area:
   - Type in a description for the alert.
   - From **Based On**, select **VM and Instance**.
   - From **What to Evaluate**, select **Hardware Reconfigured**.
   - In **Notification Frequency**, select how often you want to be notified if hardware reconfiguration is detected.
4. From **Hardware Attribute**, select Number of CPUs. From the next dropdown, select **Decreased**.

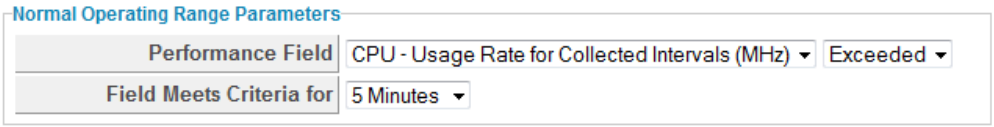


5. After setting the parameters, select what you want the alert to do. You can send an email, create an SNMP Trap, or send a management event to start an automation process.
6. Click **Add**.

Creating a Normal Operating Range Alert

Normal operating range alerts enables you to be notified when the normal operating range is exceeded, or falls below for a period of time from 1 minute to 2 hours. Capacity and utilization must be enabled for normal operating ranges to be calculated. See *General Configuration* for more information.

1. Navigate to **Control > Explorer**.
2. Click the **Alerts** accordion, then click (**Configuration**), (**Add a new Alert**).
3. In the **Info** area:
   - Type in a **Description** for the alert.
   - From **Based On**, select **VM and Instance**.
   - For **What to Evaluate**, select **Normal Operating Range**.
   - In **Notification Frequency**, select how often you want to be notified if the performance threshold is reached.
4. Set the threshold in the **Normal Operating Range Parameters** area.

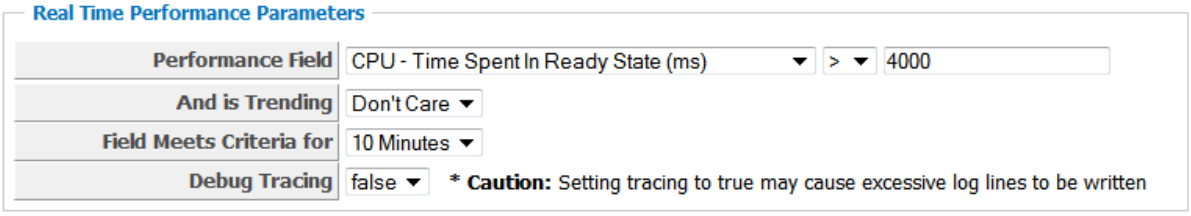


   - From **Performance Field**, select the field to check and whether you want to be notified if the field is exceeded or fell below.
   - In **Field Meets Criteria for**, select the amount of time that the threshold requires to be met to trigger the alert.
5. After setting the parameters, you then select what you want the alert to do. You can send an email, create an SNMP Trap, or send a management event to start an automation process. See Creating an Alert.
6. Click **Add**.

Creating a Real Time Performance Alert

Real Time Performance alerts enables you to be notified immediately when a performance threshold has been met for a virtual machine, host, or cluster. Capacity and Utilization must be enabled for performance thresholds to be detected.

1. Navigate to **Control > Explorer**.
2. Click the **Alert** accordion, then click (**Configuration**), (**Add a new Alert**).
3. In the **Info** area:
   - Type in a **Description** for the alert.
   - From **Based On**, select **VM and Instance**.
   - For **What to Evaluate**, select **Real Time Performance**.
   - In **Notification Frequency**, select how often you want to be notified if the performance threshold is reached.
4. Set the threshold in the **Real Time Performance Parameters** area.



   - From **Performance Field**, select the field to check and any other parameters required for that field.

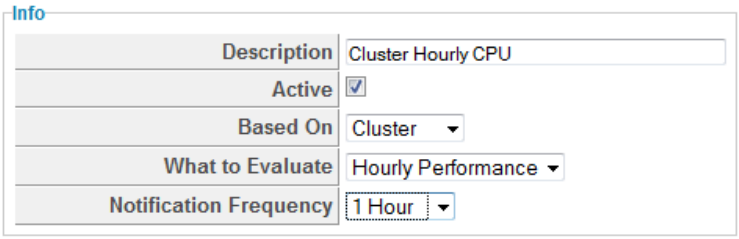
- In **And is Trending**, select **Don't Care** if it does not matter how the performance metric is trending. Otherwise, choose from the possible trending options.
- In **Field Meets Criteria for**, select the amount of time that the threshold requires to be met to trigger the alert.
- Set **Debug Tracing** to true only when directed to do so by Red Hat Support. This provides an extremely detailed level of logging and can result in many more log lines being written.

5. After setting the parameters, you then select what you want the alert to do. You can send an email, create an SNMP Trap, or send a management event to start an automation process.
6. Click **Add**.

Creating an Hourly Performance Alert

Hourly performance alerts enable you to be notified immediately when an hourly performance threshold has been met for a cluster. Capacity and Utilization must be enabled for performance thresholds to be detected.

1. Navigate to **Control > Explorer**.
2. Click the **Alerts** accordion.
3. Click (**Configuration**), (**Add a new Alert**).
4. In the **Info** area:



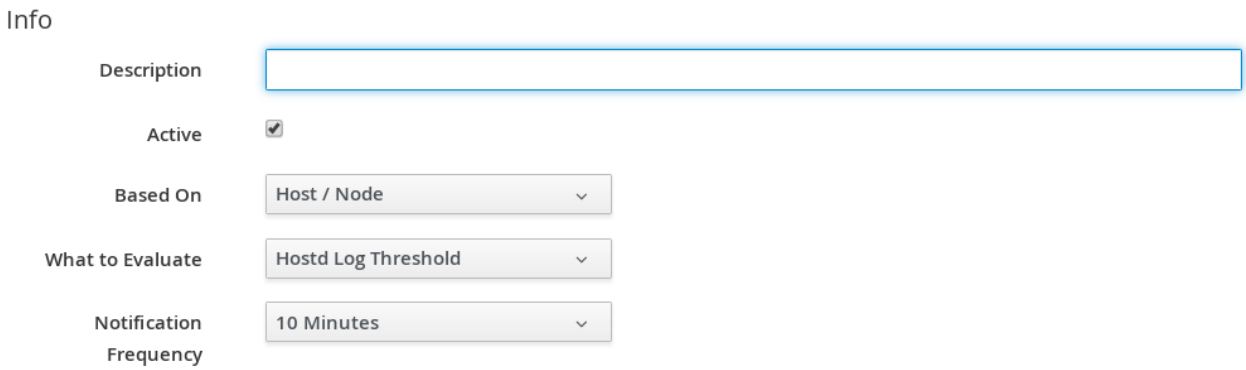
- Type in a **Description** for the alert.
- From **Based On**, select **Cluster**.
- For **What to Evaluate**, select **Hourly Performance**.
- In **Notification Frequency**, select how often you want to be notified if threshold is met.

5. In the **Hourly Performance Parameters** area select performance field and the criteria. You can also select options from the **And is Trending** dropdown box and whether the **Debug Tracing** is true or false.
6. After setting the parameters, you then select what you want the alert to do. You can send an email, create an SNMP Trap, or send a management event to start an automation process.
7. Click **Add**.

Creating a hostd Log Threshold Alert

Use the **hostd Log Threshold** to send a notification when certain items are found in the event logs for a host. A default analysis profile with event log items is required for this feature. The following example shows steps to check the host's log for a failure to validate a virtual machine's IP address.

1. Navigate to **Control > Explorer**.
2. Click the **Alert** accordion.
3. Click (**Configuration**), (**Add a new Alert**).
4. In the **Info** area:



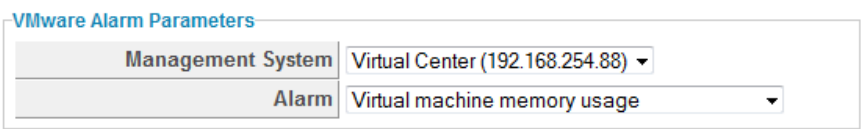
- Type in a **Description** for the alert.
- From **Based On**, select **Host/Node**.
- For **What to Evaluate**, select **Hostd Log Threshold**.
- In **Notification Frequency**, select how often you want to be notified if the log item is detected.

5. In the **Hostd Log Threshold Parameters** area, select the parameters for the event log message. You can set a threshold for a filter, level, or message source.
   - Use **Message Filter** to look for specific text in a message. Use **Message Level** to filter based on message level. Infrastructure Automation reports on the specified level and above. Use **Message Source** to filter log messages based on its source.
   - Set **How Far Back to Check** in days you want to look for this message.
   - If you only want an alert triggered when the log message has occurred a certain number of times, type the number in **Event Count Threshold**.
6. After setting the parameters, select what you want the alert to do. You can send an email, create an SNMP Trap, or send a management event to start an automation process.
7. Click **Add**.

Creating a VMware Alarm Alert

Infrastructure Automation can use VMware alarms as a trigger for an alert. This type of alert can be created for a cluster, host, or virtual machine.

1. Navigate to **Control > Explorer**.
2. Click the **Alerts** accordion, then click (**Configuration**), (**Add a new Alert**).
3. In the **Info** area:
   - Type in a description for the alert.
   - From **Based On**, select **Cluster**, **Host**, or **VM**.
   - For **What to Evaluate**, select **VMware Alarm**.
   - In **Notification Frequency**, select how often you want to be notified if the log item is detected.
4. In the **VMware Alarm Parameters** area select the provider and alarm.

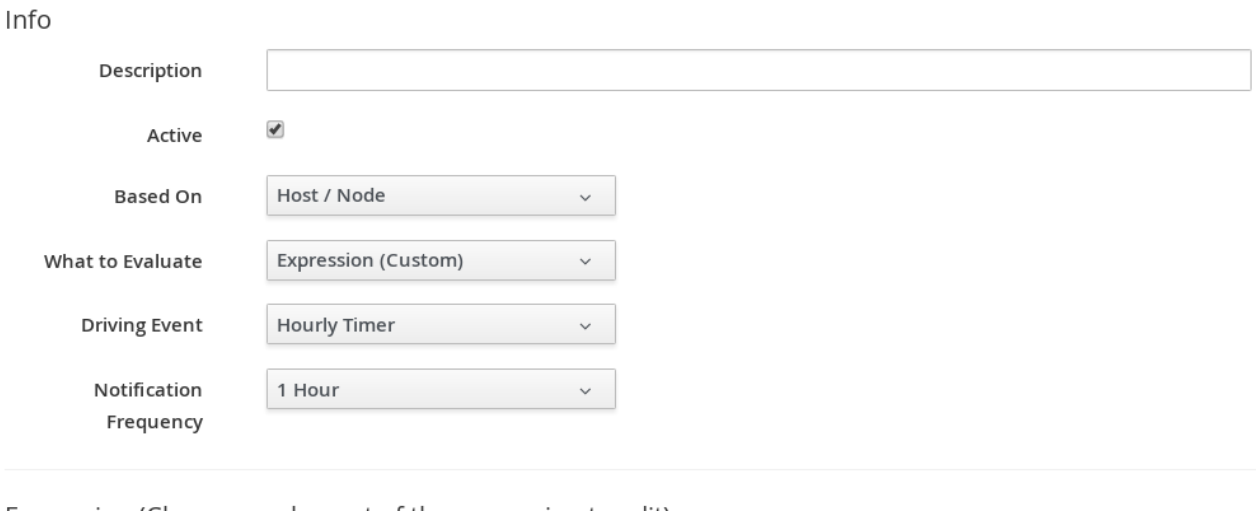


5. After setting the parameters, you then select what you want the alert to do. You can send an email, create an SNMP Trap, or send a management event to start an automation process.
6. Click **Add**.

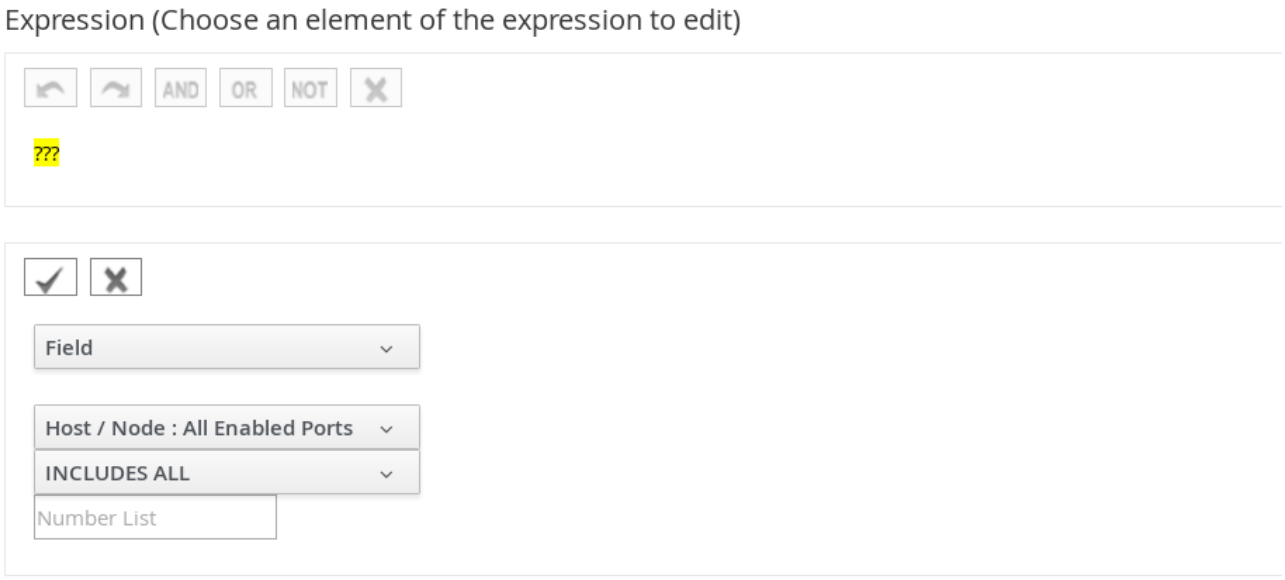
Creating an Expression Alert

Expression alerts enables you to create a notification based on any possible criteria for clusters, datastores, hosts, and virtual machines. The following procedure creates an alert for when a host's datastore has less than 5% free space.

1. Navigate to **Control > Explorer**.
2. Click on the **Alerts** accordion, then click (**Configuration**), (**Add a new Alert**).
3. In the **Info** area:



   - Type in a description for the alert.
   - From **Based On**, select **Host/Node**.
   - For **What to Evaluate**, select **Expression (Custom)**.
   - In **Notification Frequency**, select how often you want to be notified if the expression is evaluated to true.
4. Use the expression editor to create your expression. This is the same expression editor used to create **Conditions**.



5. Click (**Commit expression element changes**) to accept the expression.
6. After setting the parameters, you then select what you want the alert to do. You can send an email, create an SNMP Trap, or send a management event to start an automation process.
7. Click **Add**.

Create an Operational Alert

1. Navigate to **Control > Explorer**.

2. Click on the **Alerts** accordion, then click (**Configuration**), (**Add a new Alert**).
3. In the **Info** area:
    - Type in a description for the alert.
    - Check **Active** when you feel that the alert is ready to be enabled.
    - From **Based On**, select **Server**.
    - Select the appropriate driving event.
    - In **Notification Frequency**, select how often you want to be notified if the event log threshold is reached.
4. After setting the parameters, select what you want the alert to do. You can send an email, create an SNMP Trap, or send a management event to start an automation process.
5. Click **Add**.

Operational Alert Types

Operational Alerts

Driving Event	Explanation (Thresholds, Description)	Proposed Action and Next Steps
EVM Server Start	Alert is raised when an server starts.	Send e-email. This is a notification.
EVM Server Stop	Alert is raised when a server stops.	Send e-mail. Review logs to see why the server stopped.
EVM Server Not Responding	Alert is raised when one server detects that another server has not responded in 2 minutes.	This is a sign of a problem that should be investigated. Check logs.
EVM Server Exceeded Memory Limit	Alert is raised when an server has exceeded its system memory limit and begins killing workers. Default is 80%. Threshold configured in Advanced Settings. server: :worker_monitor: :kill_algorithm: :name: :used_swap_percent_gt_value :value: 80	This may be caused by the following issues: The server is running with too few resources. The server is enabled with too many roles or number of workers. The server picked up all the roles because another server has failed. A runaway process has taken up most of the memory.
EVM Server is Master	When one server takes over as a master server.	Typically, this should only occur when first starting a set of servers, perhaps following expected outages. If a server picks up as master in other situations, the old master had an issue that needs to be researched (such as server not responding in time).
EVM Server High System Disk Usage	The server's system disk is 80% full. This check is run as part of a system schedule. Threshold configured in Advanced Settings. server: events: :disk_usage_gt_percent: 80	Temp files used by the operating system may be filling the disk. Yum updates, normal /tmp files, or temp files in /var/lib/data/miqtemp/ may cause the problem.
EVM Server High App Disk Usage	The server's app disk is 80% full. This check is run as part of a system schedule. Threshold configured in Advanced Settings. server: events: :disk_usage_gt_percent: 80	Server temp files may remain.

Driving Event	Explanation (Thresholds, Description)	Proposed Action and Next Steps
EVM Server High Log Disk Usage	The server's log disk is 80% full. This check is run as part of a system schedule. Threshold configured in Advanced Settings. server: events: :disk_usage_gt_percent: 80	Logs are getting too big or are not being log rotated properly every day. Review the most recent logs.
EVM Server High DB Disk Usage	The server's db disk is 80% full. This check is run as part of a system schedule. Applies if using PostgreSQL as the VDMB. Threshold configured in Advanced Settings. server: events: :disk_usage_gt_percent: 80	Database or database logging is getting too large. May require **full** vacuuming of PostgreSQL database.
EVM Worker Started	Alert is raised when a worker is about to start.	This is a notification. Failover may trigger this alert.
EVM Worker Stopped	Alert is raised when a worker is requested to stop.	Review logs for reason in the event the worker was not purposefully stopped.
EVM Worker Killed	Alert is raised when a non- responsive worker does not restart on its own and is killed.	Review logs for reason the worker was killed. May be the result of EVM Worker Not Responding.
EVM Worker Not Responding	Alert is raised when a worker has not responded for 2 minutes (:heartbeat_timeout) or has not started within 10 minutes (:starting_timeout).	An influx of events from the Virtual Center or Red Hat Virtualization causes an inability of EVM/CFME to handle the capacity at which they are being queued. Utilize the Event Handler Configuration to filter events that are causing problematic queue table growth.
EVM Worker Exceeded Memory Limit	Alert is raised when a worker exceeds the memory threshold. The default is 150 MB, but some workers have their own value in the :memory_threshold section for that specific worker.	Review logs for reason the worker is exceeding the memory limit. This may be the result of an overload to the worker process that requires further investigation.
EVM Worker Exceeded Uptime Limit	Alert is raised when a worker has been running longer than the :restart_interval. (Most workers are set to never restart using the 0.hours setting.) The EMS Refresh SmartProxy workers are set to restart every 2 hours.	Review logs for the reason the worker is exceeding the limit. This may be the result of an overload to the worker process that needs further investigation.
EVM Worker Exit File	Alert is raised when the scheduler worker exits due to a pending large ntp time change.	This is a notification.

Editing an Alert

After creating an alert, you can edit the threshold, expression, or the notification type.

1. Navigate to **Control > Explorer**.
2. Click on the **Alerts** accordion, then click on the alert that you need to edit.
3. Click **(Configuration)**, **(Edit this Alert)**.
4. Make the required changes.
5. Click **Save**.

Copying an Alert

You can copy an existing alert to create a new alert that is similar to the existing one, then change the values associated with it.

1. Navigate to **Control > Explorer**.
2. Click on the **Alert** accordion, then click on the alert that you want to copy.
3. Click (**Configuration**), (**Copy this Alert**). Click **OK** to confirm.
4. Make the required changes.

5. Click **Add**.

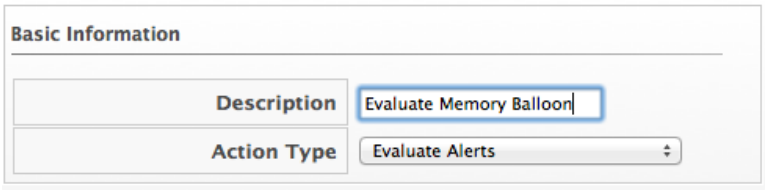
Deleting an Alert

When an alert is no longer needed, you can remove it from your VMDB.

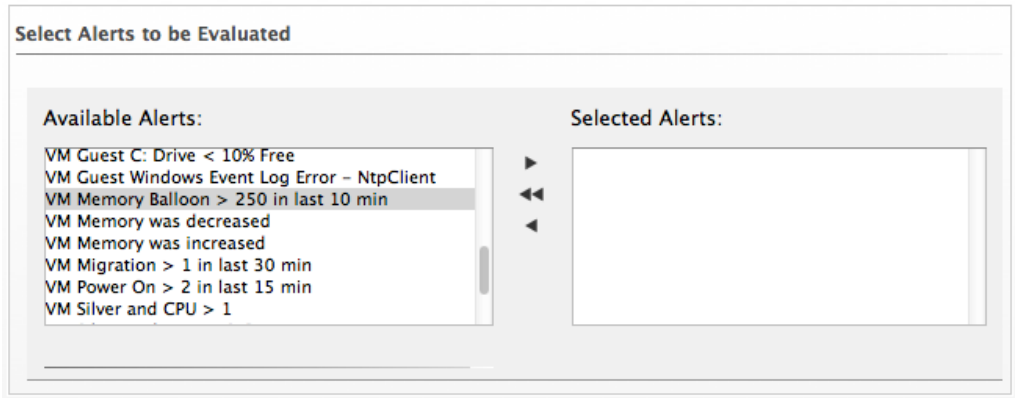
1. Navigate to **Control > Explorer**.
2. Click on the **Alerts** accordion, then click on the alert that you want to delete.
3. Click (**Configuration**), (**Delete this Alert**).
4. Click **OK** to confirm.

Evaluating an Alert

1. Navigate to **Control > Explorer**.
2. Click the **Actions** accordion, then click (**Configuration**), (**Add a new Action**).
3. Type in a **Description** for the action.



4. Select **Evaluate Alerts** from **Action Type**.
5. Select the alerts to be evaluated and click (**Move selected Alerts into this Action**). Use the `Ctrl` key to select multiple alerts.



6. Click **Add**.

Alert Profiles

Creating Alert Profiles

Alert profiles enable you to create groups of standard alerts. An alert profile can have as many alerts assigned as you need, and can be assigned to clusters, datastores, hosts, and virtual machines.

Note:

Each profile type under the **Alert Profile** accordion menu will have its own **Alert Selections** and subsequent **Assignments** choices.

1. Navigate to **Control > Explorer**.
2. Click on the **Alert Profiles** accordion, then click on the type of profile that you want to create.
3. Click (**Configuration**), (**Add a new Profile**).
4. In the **Basic Information** box, type in a unique **Description** for the alert profile.
5. Select the desired alerts from the **Alert Selection** area. Use the `Ctrl` key to select multiple alerts.

6. Click to add the Alerts.
7. Type in any additional description in the **Notes** area.
8. Click **Add**.

Editing an Alert Profile

You can edit an alert profile as your enterprise's needs change.

1. Navigate to **Control > Explorer**.
2. Click on the **Alert Profiles** accordion, then click the alert profile you want to edit.
3. Click (**Configuration**), (**Edit this Alert Profile**).
4. Make the required changes.
5. Click **Save**.

Deleting an Alert Profile

Remove alert profiles that you no longer need. This does not remove the alerts associated with the alert profile.

1. Navigate to **Control > Explorer**.
2. Click on the **Alert Profiles** accordion, then click the alert profile you want to remove.
3. Click (**Configuration**), (**Delete this Alert Profile**).
4. Click **OK** to confirm.

Assigning an Alert Profile

After an alert profile is created and verified, you can assign it directly to a resource.

1. Navigate to **Control > Explorer**.
2. Click on the **Alert Profiles** accordion, then click on the alert profile that you want to assign.
3. Click (**Configuration**), (**Edit Assignments for this Alert Profile**).
4. The options presented change based on if the alert is for a cluster, datastore, Infrastructure Automation server, host, or virtual machine and instance. You can assign to the enterprise, to specific hosts, cluster, resource pools, and providers, or based on assign tags. For a Infrastructure Automation server alert profile, you can only assign to Infrastructure Automation servers in the current Region.
5. Click **Save**.

Importing and Exporting

Importing and Exporting Policies, Policy Profiles, and Alerts

If you have multiple VMDBs, you can export policies, policy profiles, or alerts from one to another. You can export and import for use with other Infrastructure Automation infrastructures.

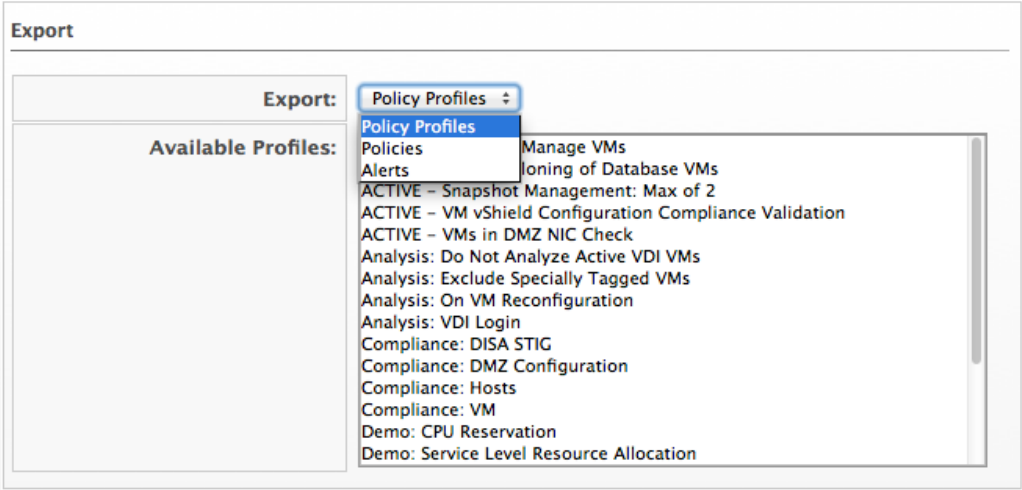
Note:

The appliances you are exporting or importing data between must use the same Infrastructure Automation version.

1. Copy the file to import to a location that is accessible to your Infrastructure Automation console.
2. Navigate to **Control > Import/Export**.
3. Click **Choose file** to navigate to the location of the file.
4. Select the file, and then click **Open** from the file selection box.
5. Click **Upload**.
6. Verify that these are the policies or policy profiles that you want to import.
7. Click **Commit**.

Exporting a Policy, Policy Profile, or an Alert

1. Navigate to **Control > Import/Export**.
2. From the **Export dropdown**, select policy profiles, policies, or alerts, depending on what you want to export.



3. From the **Available Profiles** or **Available Policies** or **Available Alerts** list, select the items to export. Use the `Ctrl` key to select multiple items to export into one file.
4. Click **Export**.
5. Follow the prompts in your browser to save the file.

OpenShift Prometheus Alerts

Note:

This feature is currently available as a technology preview only. For more information on the support scope for features marked as technology preview, see Technology Preview Features Support Scope.

Prometheus is used as an external alerting component. Infrastructure Automation integrates Prometheus alerts for use with OpenShift Container Platform. Infrastructure Automation collects events from Prometheus, generates alerts based on these events, and then attaches alerts to inventory objects.

It is possible to view ongoing alerts in Infrastructure Automation by navigating to **Monitor > Alerts** and manage their life cycle, including the ability to:

- View active alerts per provider in the **Overview** screen.
- View alert data and related objects.
- Comment, assign, and acknowledge alerts in the **All Alerts** screen.

Network utilization metrics for OpenShift Container Platform are only collected for `eth0` network interfaces. Therefore, network utilization trend will not be displayed for network interfaces other than `eth0` on the overview screen of the provider.

Configuring Prometheus Alerts in Infrastructure Automation

Configuring and enabling Prometheus alerts in Infrastructure Automation comprises the following steps:

1. Deploying and configuring Prometheus on OpenShift.
2. Assigning Prometheus alert profiles to the enterprise in Infrastructure Automation.
3. Adding an OpenShift Containers Provider with a Prometheus alert endpoint in Infrastructure Automation.

Deploying and Configuring Prometheus on OpenShift

1. Deploy Prometheus on OpenShift by following the steps covered in Prometheus on OpenShift Container Platform.
2. Add Prometheus to an OpenShift cluster and configure alert definitions. See the example code block below for configuring alerts (currently must be done on the Prometheus side):

```
$ oc edit configmap -n openshift-metrics prometheus
# Supported labels:
# severity: ERROR|WARNING|INFO. defaults to ERROR.
# Supported annotations:
# description: The error message displayed on the screen
# miqTarget: ContainerNode|ExtManagementSystem, defaults to ContainerNode.
# miqIgnore: "true|false", should ManageIQ pick up this alert, defaults to true.
  prometheus.rules: |
    groups:
    - name: example-rules
      interval: 30s # defaults to global interval
      rules:
      - alert: "NodeDown" << Alert starts here
        expr: up{job="kubernetes-nodes"} == 0
        annotations:
          miqTarget: "ContainerNode"
          severity: "ERROR"
          url: "https://www.example.com/node_down_fixing_instructions"
          description: "Node  is down"
      - alert: "TooManyRequests"
        expr: rate(authenticated_user_requests[2m]) > 12
        annotations:
          miqTarget: "ExtManagementSystem"
          severity: "ERROR"
          url: "https://www.example.com/too_many_requests_fixing_instructions"
          description: "Too many authenticated requests"
```

3. Reload the Prometheus configuration. You can do this by deleting the Prometheus pod, then redeploying with the new configuration.

Assigning Prometheus Alert Profiles to the Enterprise

Complete the following procedure to assign Prometheus alert profiles to the enterprise using the Infrastructure Automation user interface.

Note:

Both Node and Provider alert profiles are created automatically during the installation, so it is not required to create these profiles.

1. Navigate to **Control > Explorer**, then click **Alert Profiles** in the accordion menu.
2. Click to expand **Node Alert Profiles**, then click **Prometheus Node Profile**.
3. Click (**Configuration**), then click (**Edit assignments for this Alert Profile**). Assign the profile to **The Enterprise** from the **Assign To** list.
4. Click **Save**.
5. Click to expand **Provider Alert Profiles**, then click **Prometheus Provider Profile**.
6. Click **Configuration**, then click **Edit assignments for this Alert Profile**. Assign the profile to **The Enterprise** from the **Assign To** list.
7. Click **Save**.

   Note:

   Alerts triggered before assigning the profiles to the enterprise will not appear in Infrastructure Automation at all.

Adding an OpenShift Containers Provider with a Prometheus Alert Endpoint

Complete the following procedure to add an OpenShift Containers Provider with a Prometheus Alert Endpoint using the Infrastructure Automation user interface.

1. Browse to menu: **Compute > Containers > Providers**.
2. Click **Configuration**, then click (**Add a New Containers Provider**).
3. Enter a **Name** for the provider.
4. From the **Type** list, select **OpenShift Container Platform**.
5. Enter the appropriate **Zone** for the provider. If you do not specify a zone, it is set to **`default`**.
6. In the **Default** tab, configure the following for the OpenShift provider:
   1. Select a **Security Protocol** method to specify how to authenticate the provider:
      - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.

- **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.

  Note:

  You can obtain your OpenShift Container Platform provider's CA certificate for all endpoints (default, metrics, alerts) from **`/etc/origin/master/ca.crt`**. Paste the output (a block of text starting with **`-----BEGIN CERTIFICATE-----`**) into the **Trusted CA Certificates** field.

- **SSL without validation**: Authenticate the provider insecurely (not recommended).

2. Enter the **Hostname** (or IPv4 or IPv6 address) of the provider.

   Important:

   The **Hostname** must use a unique fully qualified domain name.

3. Enter the **API Port** of the provider. The default port is **`6443`**.

4. Enter a token for your provider in the **Token** box.

   Note:

   To obtain a token for your provider, run the **`oc get secret`** command on your provider; see Obtaining an OpenShift Container Platform Management Token.

   For example:

   # oc get secret –namespace management-infra management-admin-token-8ixxs –template='' | base64 –decode

5. Click **Validate** to confirm that Infrastructure Automation can connect to the OpenShift Container Platform provider.

7. In the **Metrics** endpoint tab optionally configure the metrics service details:

   1. Select a service type (e.g.: Prometheus) or leave disabled

   2. Select a **Security Protocol** method to specify how to authenticate the service:

      - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.

      - **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.

      - **SSL without validation**: Authenticate the provider insecurely using SSL. (Not recommended)

   3. Enter the **Hostname** (or IPv4 or IPv6 address) of the metrics service.

      In order to find the hostname you can use the **`oc get route`** command.

      You can retrieve the Prometheus route by running:

      ```
      oc get route prometheus-k8s -n openshift-monitoring
      ```

      Example:

      ```
      oc get route prometheus-k8s -n openshift-monitoring
      NAME              HOST/PORT                                                PATH              SERVICES
      PORT      TERMINATION          WILDCARD
      prometheus-k8s   prometheus-k8s-openshift-monitoring.ocp.example.com  prometheus-k8s   web
      reencrypt/Redirect   None
      ```

      In this example you would use **`prometheus-k8s-openshift-monitoring.ocp.example.com`** as the metrics hostname.

   4. Enter the **API Port** of your metrics endpoint uses a non-standard port for access. The default port is **`443`**.

   5. Click **Validate** to confirm that Infrastructure Automation can connect to the metrics endpoint.

8. In the **Virtualization** endpoint tab optionally configure the virtualization service details:

   1. Select a service type (OpenShift Virtualization / KubeVirt) or leave disabled

   2. Select a **Security Protocol** method to specify how to authenticate the service:

      - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.

- **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.
- **SSL without validation**: Authenticate the provider insecurely using SSL. (Not recommended)

3. Enter the **Hostname** (or IPv4 or IPv6 address) of the virtualization service.
4. Enter the **API Port** of your virtualization service. The default port is **`443`**.
5. Enter a token for your provider in the **Token** box.

9. Click the **Advanced** tab to add image inspector settings for scanning container images on your provider using OpenSCAP.

Note:

- These settings control downloading the image inspector container image from the registry and obtaining the Common Vulnerabilities and Exposures (CVE) information (for effective scanning) via a proxy.
- CVE URL that Infrastructure Automation requires to be open for OpenSCAP scanning: https://www.redhat.com/security/data/metrics/ds/. This information is based on the source code of OpenSCAP.

1. Enter the proxy information for the provider in either **HTTP**, **HTTPS**, or **NO Proxy** depending on your environment.
2. Enter the **Image-Inspector Repository** information. For example, **`openshift3/image-inspector`**.
3. Enter the **Image-Inspector Registry** information. For example, **`registry.access.redhat.com`**.
4. Enter the **Image-Inspector Tag** value. A tag is a mark used to differentiate images in a repository, typically by the application version stored in the image.
5. Enter **`https://www.redhat.com/security/data/metrics/ds/`** in **CVE location**.

10. Click **Add**.

Note:

You can also set global default image-inspector settings for all OpenShift providers in the advanced settings menu by editing the values under **`ems_kubernetes`**, instead of setting this for each provider.

For example:

```
:image_inspector_registry: registry.access.redhat.com
:image_inspector_repository: openshift3/image-inspector
```

Once you have completed the procedure, you will have OpenShift Prometheus alerts enabled in Infrastructure Automation, and can manage their life cyle from the **Monitor > Alerts** screen in the user interface.

Regular Expressions

In Infrastructure Automation, regular expressions can be used to search the contents of a file for a specific string for use in a condition. Below are listed the items most commonly used with Infrastructure Automation to search strings. These are a small subset of all the items available to use in regular expressions. If you are unfamiliar with regular expressions, there are many resources available on the Internet, including www.regular-expressions.info. Note that if you want to search a file, you must collect it as part of a host analysis profile.

Anchors		
^	start of string	
$	end of string	
Character Classes		
\s	white space including spaces, tabs, and line breaks	
\S	not white space	
\d	digit, same as [0-9]	
\D	not digit	
\w	word	
\W	not word	
Quantifiers		
*	0 or more of preceding characters	

+	1 or more of preceding characters	
?	0 or 1 of preceding character	
Escape Character		
\	put before a metacharacter to search for that actual character	
Metacharacters		
^ [. $ { * (\ +)	? < >	must be used with the Escape Character if you are searching specifically for it
Special characters		
\n	new line	
\t	tab	
Groups and Ranges		
.	any character except new line (\n)	
(a	b)	a or b
()	group	
[abc]	a or b or c	
[^abc]	not a or b or c	
[a-q]	letter between a and q	
[A-Q]	upper case letter between A and Q	
[0-7]	digit between 0 and 7	
Pattern modifiers		
i	case insensitive	
Other helpers		
.*	swallows text between 2 words	
\s+	guarantees minimum of 1 whitespace between 2 words	
\s*	guarantees 0 or more whitespace between 2 words	
^\s*	beginning of line with zero or more whitespace	
\s+.*	swallows all text and white space between 2 words	
\d+	guarantees minimum of 1 number between 2 words	
<\w>	identical to <[a-zA-Z0-(_]>	

Regular Expressions

Description	Regular Expression			
([a-zA-Z0-9]+)	Letters, numbers, hyphens			
Find the line beginning with **sshd**. Then, using a colon `:` as delimiter, check that the value four ":" over is equal to 99999.	^sshd:[^:]*:[^:]*:[^:]*:99999:			
Verify that **PASS_MAX_DAYS** exists starting in position `1` and a value after it is ⇐ 90.	^\s*PASS_MAX_DAYS\s+([0-9]	[1-8][0-9]	90)	
Verify that **ROOTPW** (in any case) exists on an uncommented line.	/[#]*ROOTPW/i			
Verify that line in file starts with size and the value after is ⇐ 4096k.	^\s*size\s+(409[0-6]	40[0-8][0-9]	[123][09]{3}	\d{1,3})k
Find line with string **restrict 127.0.0.1** that starts in position `1 to ensure it is not commented out.	^\s*restrict\s+127\.0\.0\.1			
Find an uncommented line that contains **/home**. There will be additional text before the desired string.	[#]*\/home			

Examples

Reportable Fields in Infrastructure Automation

Infrastructure Automation provides a set of out of the box reports covering a variety of areas. These fields are collected by Infrastructure Automation from the different parts of the infrastructure. Most of the fields are named directly as the vendor would name them. However, there

are some fields that Infrastructure Automation calculates given this data. Note that this list may vary by Provider and may change as new Providers are added to Infrastructure Automation. Also, note that the value returned by each reportable field below is in megabytes (MB).

Reportable Field
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg

Clusters

Reportable Field
Datastore : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Datastore : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Datastore : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
VM Template : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
VM Template : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
VM Template : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
VM Template : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VM and Instance : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VM and Instance : Memory - Avg Used for Collected Intervals 30 Day Avg
VM and Instance : Memory - Avg Used for Collected Intervals 30 Day High Avg
VM and Instance : Memory - Avg Used for Collected Intervals 30 Day Low Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.File System.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
File Share.File System.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
File Share.File System.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.File System.Logical Disk.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
File Share.File System.Logical Disk.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
File Share.File System.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.File System.Logical Disk.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
File Share.File System.Logical Disk.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
File Share.File System.Logical Disk.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.File System.Logical Disk.VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
File Share.File System.Logical Disk.VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
File Share.File System.Logical Disk.VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
File Share.File System.Logical Disk.VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg
File Share.File System.VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
File Share.File System.VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
File Share.File System.VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
File Share.VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg

Reportable Field
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg

Datastore Files

Reportable Field
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.File System.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
File Share.File System.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
File Share.File System.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.File System.Logical Disk.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
File Share.File System.Logical Disk.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
File Share.File System.Logical Disk.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.File System.Logical Disk.VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
File Share.File System.Logical Disk.VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
File Share.File System.Logical Disk.VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
File Share.File System.Logical Disk.VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg
File Share.File System.VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
File Share.File System.VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
File Share.File System.VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
File Share.File System.VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg
File Share.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
File Share.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
File Share.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
File Share.VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
File Share.File System.VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
File Share.File System.VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg
File Share.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
File Share.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
File Share.Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
File Share.VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
File Share.VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
File Share.VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
File Share.VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Hosts : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg

Datastores

Reportable Field
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg

Reportable Field
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg

EVM Groups

Reportable Field
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Cluster : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Cluster : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Cluster : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Miq Templates : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg

Hosts

Reportable Field
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
Cluster : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Cluster : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Cluster : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Datastore : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Datastore : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Datastore : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Datastores : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Host : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Host : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Host : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg

Reportable Field
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Provider : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Provisioned From Template : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Provisioned From Template : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Provisioned From Template : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Provisioned From Template : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
Provisioned VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Avg
Provisioned VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day High Avg
Provisioned VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Low Avg
Provisioned VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
Service.VMs : Memory - Aggregate Used for Child VMs for Collected Intervals 30 Day Max
Service.VMs : Memory - Avg Used for Collected Intervals 30 Day Avg
Service.VMs : Memory - Avg Used for Collected Intervals 30 Day High Avg
Service.VMs : Memory - Avg Used for Collected Intervals 30 Day Low Avg

Images

Chargeback

The Chargeback dashboard for IBM Cloud Pak for AIOps includes the utilization, reporting, and chargeback data from the Infrastructure Automation – Infrastructure Management component.

You can use the {{site.data.keyword.cpconsole}} to view and manage chargeback cost. The chargeback feature provides the way to calculate monetary virtual machine charges based on owner or company tag. Access the Chargeback dashboard by browsing to the menu, click **Manage costs > Chargeback**

- Tag Categories and Tags
- Chargeback Rates
- Chargeback Assignments
- Chargeback Reports

Tags

With Infrastructure management, you can create your own set of tags and tag categories. You can use tags to create a customized, searchable index for your resources. After you create these values, you can apply them to your resources.

Company Tag Categories and Tags

Infrastructure management provides two types of tags. Company tags and System tags.

Company tags that you see under **My Company Tags** for a resource.

1. Create company tags from the console by browsing to the menu, click **Automate infrastructure > Infrastructure management**
2. Click **Settings**, then click **Application Settings**.
3. Select the region: **Region** and click the **Tags** tab.
4. Select the **My Company Tags** tab.

A selection of company tags is provided to you by default as samples. Tags can be deleted if you don't need them, but are not re-created by Infrastructure management.

System tags are assigned automatically by Infrastructure management.

For more information, see Tags.

Chargeback Rates

Infrastructure management provides a default set of rates for calculating chargeback costs.

You can configure chargeback rates for various resources by selecting either Compute or Storage in the Rates tab. Compute sets chargeback rates for CPU, disk I/O, fixed compute cost, memory, and network I/O items. Storage sets chargeback rates for fixed storage costs and disk storage.

Chargeback costs are computed by using a set formula based on hourly cost per unit and hourly usage.

Chargeback can be calculated in many currencies.

Chargeback rates can be assigned at a single rate or by tiers, where rates are assigned in ranges that depend on the level of usage.

Creating Chargeback Rates

Infrastructure management provides the capability to create your own set of computing and storage costs to use for billing.

Chargeback can be calculated at one fixed rate, or by a combination of fixed and variable rates per tier. Fixed rates are charged once per unit of time. The variable rate is calculated by the level of usage that is multiplied by the number of resources that are used in a unit of time.

Adding a Chargeback Rate

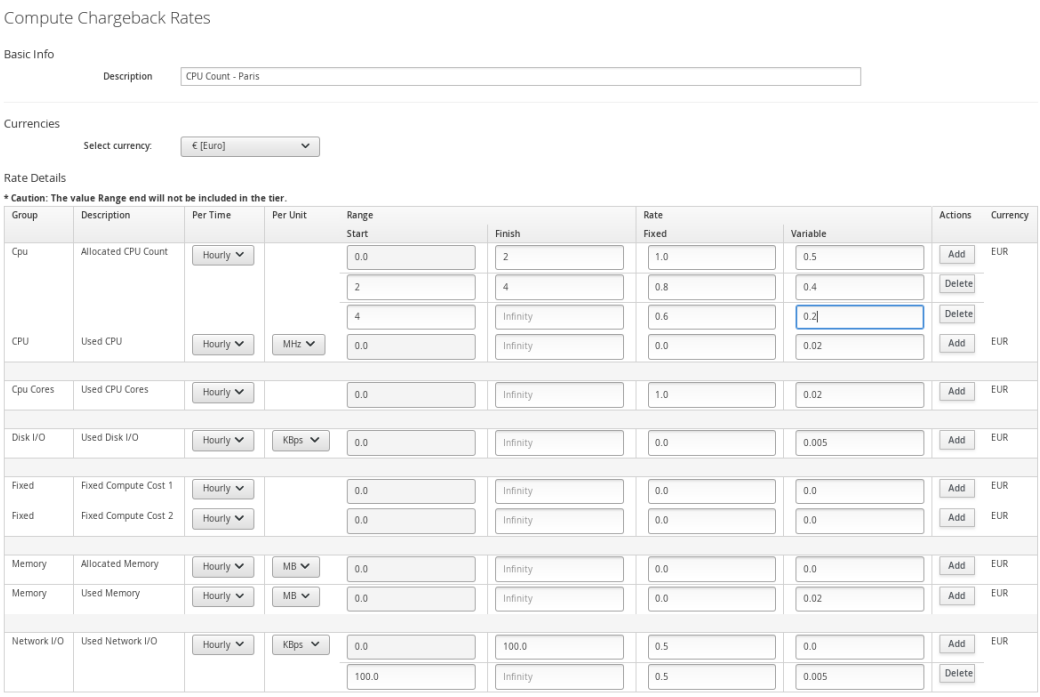
The following example shows a scenario to create a new chargeback rate for allocated CPU usage in Euro per week measured by the following tiers, calculating both fixed and variable costs:

- Tier 1: 0-2 CPU = 1 Euro fixed + 0.5 Euro variable
- Tier 2: 2-4 CPU = 0.8 Euro fixed + 0.4 Euro variable
- Tier 3: 4 CPU or more = 0.6 Euro fixed + 0.2 Euro variable

To configure this rate:

1. Browse to the menu, click **Manage costs > Chargeback**.
2. Click the **Rates** tab, then click **New rate**. The Infrastructure management is launched in a new browser tab.

   Note: You can also navigate to Chargeback rates from the Infrastructure management tab if it is already launched by selecting, **Overview** > **Chargeback** > **Rates**. Select **Configuration** > **Add a new Chargeback Rate**.
3. Enter a **Description** for the chargeback rate.
4. Confirm that the **Type** is set to **Compute**.
5. Select the currency for the new rate. For example, **Euro**.
6. In **Rate Details**, select the time, and complete the range and rates for each tier. Optionally, a single chargeback rate can be configured by using only one tier. For this example, in **Allocated CPU Count**, configure:
    1. For tier 1, set:
        1. In **Per Time**, select **Hourly**.
        2. In **Range**, specify the CPU range: **Start** remains as `0.0` and **Finish** is `2`.
        3. In **Rate**, specify `1` for the **Fixed** rate, and `0.5` for the **Variable** rate.
        4. Click **Add** under **Actions** to create a new, blank **Allocated CPU Count** line for tier 2.
    2. For tier 2, set:
        1. In the new line in **Range**, specify the CPU range: **Start** is `2` and **Finish** is `4`.
        2. In **Rate**, specify `0.8` for the **Fixed** rate, and `0.4` for the **Variable** rate.
        3. Click **Add** under **Actions** to create another new, blank **Allocated CPU Count** line for tier 3.
    3. For tier 3, set:
        1. In the new line in **Range**, specify the CPU range: **Start** is `4` and leave **Finish** as `Infinity`.
        2. In **Rate**, specify `0.6` for the **Fixed** rate, and `0.2` for **Variable**.



Tiered chargeback

7. Click **Add** to save the chargeback rate.

The rate can now be assigned.

For more information, see Chargeback Rates.

Chargeback assignments

Compute rates can be assigned to

- The enterprise
- Selected clusters
- Selected providers
- Tagged VMs and instances
- Tenants

Storage rates can be assigned to

- The enterprise
- Selected datastores
- Tagged datastores
- Tenants

Infrastructure management provides you with the capability to assign chargeback rates by choosing from **Compute** and **Storage**. You can assign either a custom chargeback rate, or a default rate for Compute, Storage, or Container Images.

Note: The **Default Container Image Rate** can be useful when applied to a container provider with multiple images, where only some of the images have a chargeback rate assigned. The images without a custom rate use the default rate.

1. Browse to the menu, click **Manage costs > Chargeback**.
2. Click the **Assignments** tab, then click **Create assignments**. The Infrastructure management component is called.
3. From the **Chargeback Assignments** menu, click **Compute** or **Storage**.

Configuring Chargeback for Compute

Use **Compute** to assign a compute chargeback rate. You can assign chargeback rates to **The Enterprise**, **Selected Providers**, **Selected Clusters**, **Tagged VMs and Instances**, **Tagged Configured Systems**, **Tagged Images**, **Labeled Images**, or **Tenants**.

1. From the **Selections** area, choose each item to set, select the chargeback rate to use. The options that are displayed vary based on the type you select.

2. For **Tagged VMs and Instances** and **Tagged Images** select a **Tag Category**.
3. Click **Save**.

Configuring Chargeback for Storage

Use **Storage** to assign a storage chargeback rate. You can assign chargeback rates to **The Enterprise**, **Selected Datastores**, **Tagged Datastores**, or **Tenants**.

1. From the **Selections** area, choose each item to set, select the chargeback rate to use. The options that are displayed vary based on the type you select.
2. For **Tagged Datastores**, select a **Tag Category**.
3. Click **Save**.

The rate is assigned. The next time that you generate a chargeback report, these values will be used.

Note: When you view chargeback, you see a rate for a virtual machine for the number of the CPUs. The chargeback for this parameter is calculated based on when the virtual machine is running. If the virtual machine isn't running, then it is not charged for CPU allocation.

For more information, see Assigning Chargeback Rates.

Chargeback Reports

Prerequisites for running chargeback reports

1. Enable Capacity & Utilization Collection and Metric Workers
    1. Browse to the menu, click **Automate infrastructure > Infrastructure management**.
    2. Click **Settings**, then click **Application Settings**.
    3. Click **Settings > Current Region**, then click the **C & U Collection** tab.
    4. Metrics data is needed to generate chargeback reports. It is required to enable collecting metrics data. You can select individual cluster(s), datastore(s) or select all clusters and datastores.
    5. Click **Save**.
2. Set Server Roles
    1. Browse to the menu, click **Automate infrastructure > Infrastructure management**.
    2. Click **Settings**, then click **Application Settings**.
    3. Click **Settings > Current Region**, then in the tree select **Zones > Current Zone > Current Server**.
    4. In the **Server Control** section turn on the required server roles to get metric data for chargeback reports:
        - Capacity & Utilization Coordinator
        - Capacity & Utilization Data Collector
        - Capacity & Utilization Data Processor
3. Set Assignments
    1. See Chargeback Assignments
    2. Assign rates to a specific scope of resources (The Enterprise, Selected Providers,...). All resources (VMs,...) have an assigned rate when you select **The Enterprise** (for Compute and Storage) in the **Assigned To** selection box.
4. Set Chargeback Filters on chargeback reports

    Filters on chargeback reports must match resources to see any results on the report result. Setting of the filters affects the scope of resources taken into account for chargeback report generation. You can display all resources (VMs,...) on the report result by setting **Show Costs by** to **tenant** and **Tenant** to **My Company** in the **Chargeback Filter** section. When creating reports, click the **Filter** tab to set filters for the data displayed in the report.

Chargeback Reports can be generated based on:

- Allocation
- Utilization

Creating a Chargeback Report

Infrastructure management provides you with the capability to create chargeback reports to monitor costs.

Note: When you create a chargeback report, the **Total Cost** is the complete cost for the chosen workload over the selected time frame. Total costs are calculated by adding the calculated costs for the amount that is used and the amount that is allocated for the time frame. When you add cost-related columns to display on your report, be sure to select all rate assigned fields to present a comprehensive accounting of total costs.

1. Browse to the menu, click **Manage costs > Chargeback**.

2. Select the **Reports** tab, then click **Create reports** or **New report**.

   The Infrastructure management is launched in a new browser tab.

3. From the Infrastructure management, select the **Overview** > **Reports**.
4. Click **Reports** accordion, then select **Configuration** and click **Add a new Report**.
5. On the **Columns** tab, complete the **Basic Report Info** area.
   - Type a unique name in **Menu Name** for how you want the report to be named in the menu list.
   - Type the **Title** to display on the report.
6. Add fields in the **Configure Report Columns** area.
   - From the **Base the report on** list, select **Chargeback for Vms**, **Chargeback for Configured Systems**, **Containers**, or any other object.
   - Select the fields to include in the report from the **Available Fields** list, then click **Move selected fields down**. Also, you can select any tags that you created and assigned.
   - Change the order of the fields in the report by clicking **Move up selected fields** or **Move down selected fields**.
7. Click the **Formatting** tab to set the size of paper for a PDF and column header format.
   - From the **PDF Output** area, select the page size from the **Page Size** list.
   - From **Specify Column Headers and Formats**, type the text to display for each field. For each numeric field, you can also set the numeric format.
8. Click the **Filter** tab to set filters for the data displayed in the report.
   - From **Chargeback Filters**, select how you want the costs to show, the tag category, the tags, and how you want the items to be grouped. **Note:** You can select multiple tags.
   - From **Chargeback Interval**, select the time interval. You must have a full interval worth of data to select an option other than **Partial** in the **Daily Ending With** list.
9. Click the **Preview** tab, and then **Load** to see how the report appears.
10. When you're satisfied with the report, click **Add** to create the new report.

The new report is created. To make the report accessible from the **Report** menu, you must add it to a report menu.

For more information, see Creating a Chargeback Report.

Running Reports

Generating a Single Report

1. Browse to the menu, click **Manage costs > Chargeback**.
2. Select the **Reports** tab, and then click the report name in the **Reports** section.
3. Click **Run report** to generate the report.

Infrastructure management uses interactive report generation so that reports are placed on a queue. A visual indicator of the reports status is shown. All reports are automatically saved so that they can be downloaded and analyzed later.

Viewing Reports

Once you have run a report, you can view it at any time after the report has been generated.

1. Browse to the menu, click **Manage costs > Chargeback**.
2. Select the **Reports** tab, and then click the report name in the **Reports** section.
3. From the **Saved Reports**, click the report **Last run** entry under the **Saved Reports** to view the report details.

Note: From the report details you can click the column headings to sort the details by value.

Exporting Reports

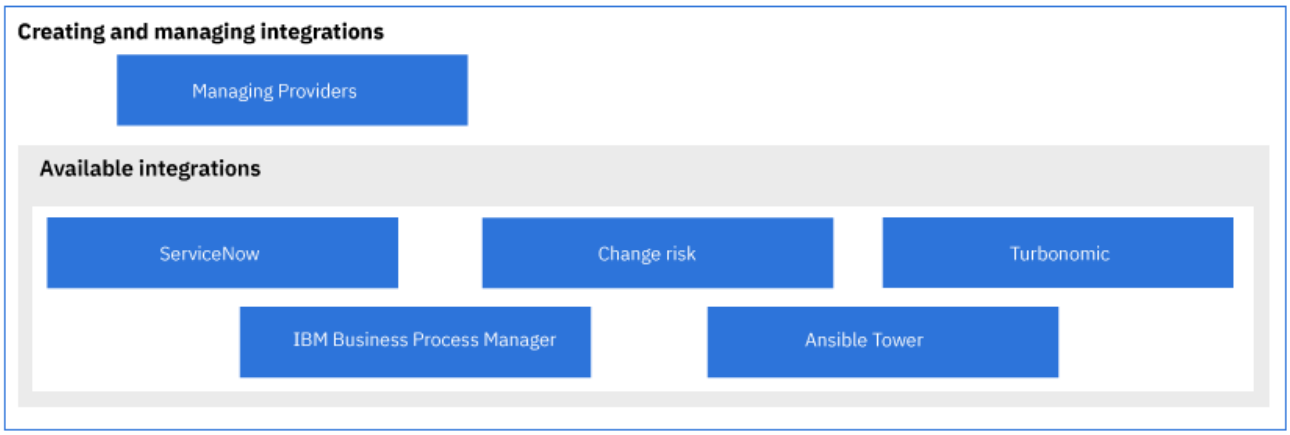
1. Browse to the menu, click **Manage costs > Chargeback**.
2. Select the **Reports** tab, and then click the report name in the **Reports** section.

3. From the **Saved Reports**, click the report **Last run** entry under the **Saved Reports** to view the report details.
4. Click **Export Report** and select **Download as Text** or **Download as CSV**.

Managing integrations

Learn more about the available integrations for Infrastructure Automation.

To learn more about managing integrations for Infrastructure Automation, click a subject within the following image to view the related available documentation (or click the subject from list that follows the image):



- Managing Providers
- ServiceNow integration
- Change risk integration
- Turbonomic integration
- Business Process Manager integration
- Ansible Tower integration

- Managing Providers
- Infrastructure Providers
- Refreshing Providers
- Tagging Multiple Providers
- Viewing a Provider
- Removing a Provider
- Viewing the Provider Timeline
- Viewing Hosts and Clusters
- Viewing Virtual Machines and Templates
- Configuration Management Providers
- Automation Management Providers
  - Embedded Workflows
    - Embedded Workflows UI
    - Adding an Embedded Workflows Repository
    - Viewing the List of Workflows in a Repository
    - Refreshing Repositories
    - Adding Credentials
      - Credential Types
        - SCM
    - Authoring a workflow
      - Running an Embedded Workflow on Appliances
      - Running an Embedded Workflow on the OpenShift Container Platform (OCP)
      - Example: Provisioning Workflow
    - Viewing workflow details
    - Creating a Service Dialog using an Embedded Workflow
    - Creating Service Catalog Items using an Embedded Workflow
  - Ansible
    - Enabling the Embedded Ansible Server Role
    - Verifying the Embedded Ansible Worker State

        - Adding a Playbook Repository
        - Refreshing Repositories
- Credentials
    - Adding Credentials
    - Credential Types
        - Machine
        - Network
        - SCM
        - Amazon
        - Azure
        - OpenStack
        - Red Hat Virtualization
        - VMware
- Tagging Ansible Playbooks, Repositories, and Credentials
    - Adding Tags to Ansible Playbooks
    - Adding Tags to Ansible Repositories
    - Adding Tags to Ansible Credentials
- Optimizing Ansible Playbooks for Infrastructure Automation
    - Installing Roles on an Embedded Ansible Appliance
    - Ansible Service Linking
        - Example: Linking a virtual machine to a service
        - Modifying the Automate Workspace Using the manageiq-automate Role.
            - Role Variables
            - Example Playbook
    - Callbacks in Multiple Appliance Environments
        - Ansible Tower
            - Working with an Ansible Tower Provider
            - Adding an Ansible Tower Provider
            - Refreshing an Ansible Tower Provider
            - Viewing Ansible Tower Providers and Inventory
            - Viewing Ansible Tower Configured Systems
            - Executing an Ansible Tower Job or Workflow Template from a Service Catalog
            - Executing an Ansible Tower Job Using a Custom Automate Button
- Cloud Providers
- Refreshing Cloud Providers
- Tagging Cloud Providers
- Removing Cloud Providers
- Editing a Cloud Provider
- Viewing a Cloud Provider's Timeline
- Physical Infrastructure Providers
- Image Import Workflows
    - IBM PowerVC to IBM Power Systems Virtual Server
        - Preparing Infrastructure Automation hosting server (required once):
        - Preparing IBM PowerVC Server (required once):
        - Add a Cloud Object Storage provider in Infrastructure Automation:
        - Add an IBM Power Systems Virtual Server provider in Infrastructure Automation:
        - Add an IBM PowerVC provider in Infrastructure Automation:
        - Grant Image Import Permissions in Infrastructure Automation:
        - Enable Embedded Ansible in Infrastructure Automation:
        - Start the workflow
        - Troubleshooting
- Network Managers
- Adding or Viewing Network Providers
- Refreshing Network Providers
- Tagging Network Providers
- Removing Network Providers
- Viewing a Network Provider's Timeline
- Using the Topology Widget for Network Providers
- Containers Providers
- Tagging Containers Providers
- Removing Containers Providers
- Editing a Containers Provider

- Hiding Environment Variables for Containers Providers
- Viewing a Containers Provider's Timeline
- Storage Managers
  - Appendix
  - Using a Self-Signed CA Certificate

Managing Providers

Infrastructure Automation can manage a variety of external environments, which are known as providers and managers. A provider or manager is any system that Infrastructure Automation integrates with for the purpose of collecting data and performing operations.

In Infrastructure Automation, a *provider* is an external virtualization, cloud, or containers environment that manages multiple virtual machines or instances residing on multiple hosts. One example is Red Hat Virtualization, a platform that manages multiple hosts and virtual machines.

In Infrastructure Automation, a *manager* is an external management environment that manages more than one type of resource. One example of a manager is OpenStack, which manages infrastructure, cloud, network, and storage resources.

This guide covers working with providers and managers in Infrastructure Automation, which include:

- Infrastructure providers
- Configuration management providers
- Automation management providers
- Cloud providers
- Physical infrastructure providers
- Networking management providers
- Container providers
- Storage managers

For information on working with the resources that are contained by a provider or manager, see Managing Infrastructure and Inventory.

Infrastructure Providers

In Infrastructure Automation, an infrastructure provider is a virtual infrastructure environment that you can add to a Infrastructure Automation appliance to manage and interact with the resources in that environment. This chapter describes the different types of infrastructure providers that you can add to Infrastructure Automation, and how to manage them. Infrastructure providers can be either discovered automatically by Infrastructure Automation, or added individually.

The web interface uses virtual thumbnails to represent infrastructure providers. Each thumbnail contains four quadrants by default, which display basic information about each provider:

1. Number of hosts
2. Management system software
3. Currently unused
4. Authentication status

Icon	Description
	Validated: Valid authentication credentials have been added.
	Invalid: Authentication credentials are invalid.

Icon	Description
?	Unknown: Authentication status is unknown or no credentials have been entered.

For information on managing specific infrastructure providers see the following.

- Red Hat Virtualization Providers
- OpenStack Infrastructure Providers
- VMware vCenter Providers
- IBM Power HMC
- OpenShift Virtualization (KubeVirt)

Refreshing Providers

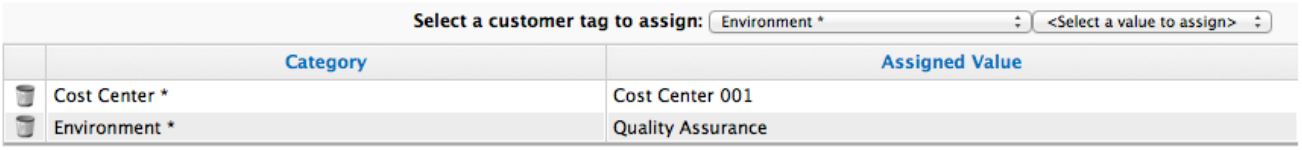
Refresh a provider to find other resources related to it. Use **Refresh** after initial discovery to get the latest data about the provider and the virtual machines it can access. Ensure the provider has credentials to do this. If the providers were added using **Discovery**, add credentials using (**Edit Selected Infrastructure Provider**).

1. Browse to menu: **Compute > Infrastructure > Providers**.
2. Select the providers to refresh.
3. Click **Configuration**, and then (**Refresh Relationships and Power States**).
4. Click **OK**.

Tagging Multiple Providers

Apply tags to all providers to categorize them together at the same time.

1. Navigate to **Infrastructure > Providers**.
2. Check the providers to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. In the **Tag Assignment** area, select a customer tag to assign from the first list, then select a value to assign from the second list.



5. Select more tags as required; click (**Save**).

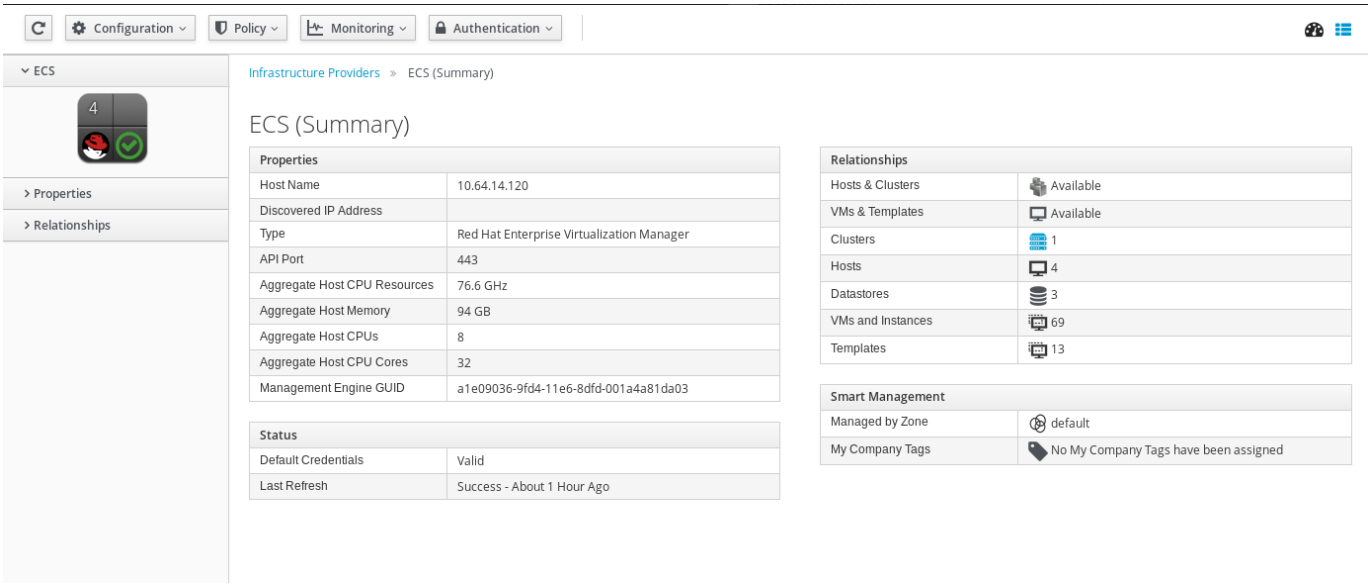
Viewing a Provider

From a list of providers, you can review a specific provider by clicking on it. This displays various options to access provider information.

There are two methods of viewing an infrastructure provider's details: the summary screen (default) and the dashboard screen. Use the summary and dashboard buttons to toggle between views.

Both the summary and dashboard screens contain a taskbar with **Reload**, **Configuration**, **Policy**, **Monitoring**, and **Authentication** buttons to manage the selected provider.

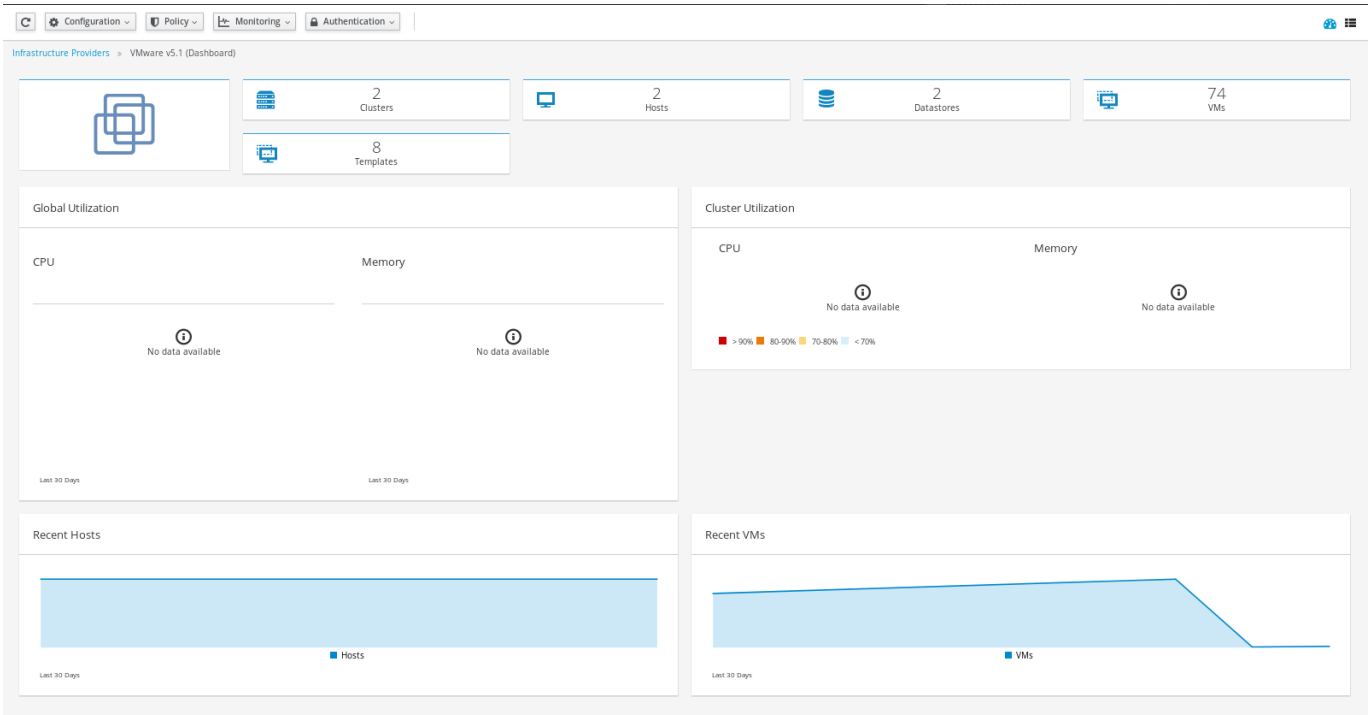
Provider Summary Screen.



The provider summary screen displays information about the provider in table format.

- Provider accordion: Displays details about the provider's **Properties** and **Relationships** on the sidebar. Click to expand these lists.
- Provider summary: Displays a provider's **Properties**, **Status**, **Relationships**, and **Smart Management**. Click on an item in the **Relationships** table to see more information about that entity.

Provider Dashboard Screen.



From the dashboard, you can view:

- Number of clusters, hosts, virtual machines, templates, datastores, resource pools, and other entities on the provider. Click on an entity to see more information about that item.
- Aggregate utilization for CPU, memory, and storage
- Network I/O statistics
- Trends for hosts and virtual machines discovered

To view the dashboard:

1. Browse to menu: **Compute > Infrastructure > Providers**.

2. Click the infrastructure provider to view.
3. To access the dashboard view, click (**Dashboard view**).

To return to the summary view, click (**Summary view**).

Removing a Provider

If a provider has been decommissioned or requires some troubleshooting, it might require deletion from the VMDB.

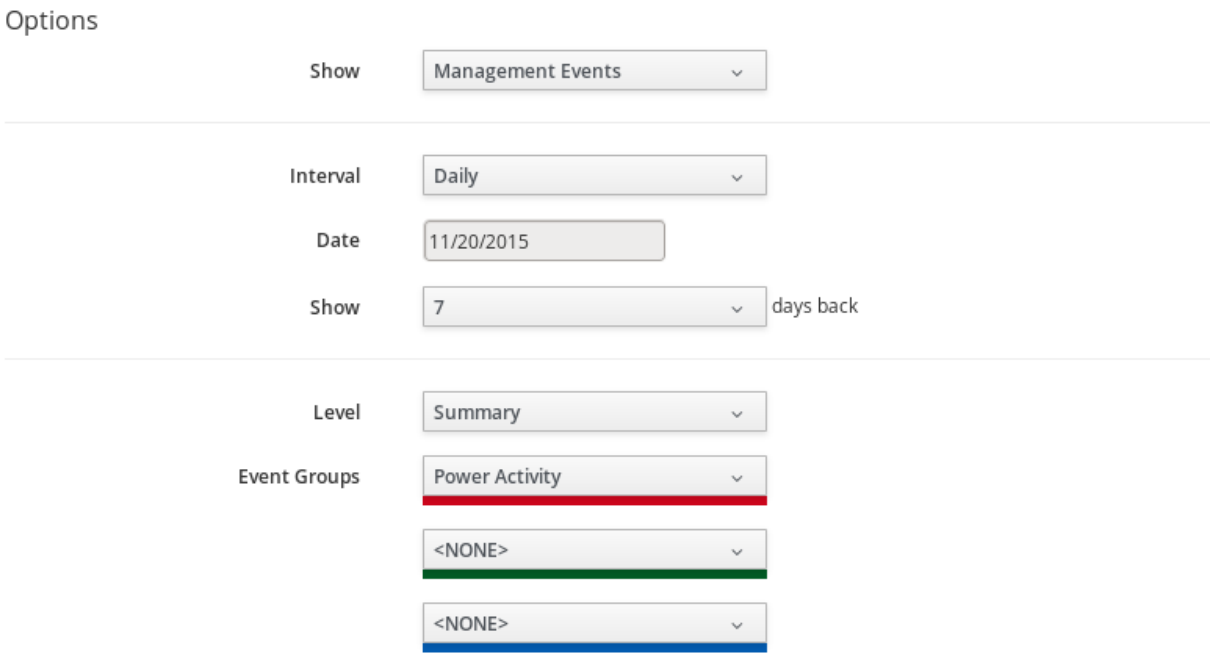
Deleting a provider removes the account information from Infrastructure Automation console. You will no longer be able to view any associated history including chargeback reports generated for the deleted provider. Additionally, if Infrastructure Automation is the database of record, deleting providers would become a major problem for the other systems relying on it for accurate and consistent billing information. Review all the dependencies carefully before deleting a provider.

1. Navigate to **Compute > Infrastructure > Providers**.
2. Select the check box for the provider to delete.
3. Click (**Configuration**), then (**Remove Infrastructure Providers from the VMDB**).
4. Click (**OK**).

Viewing the Provider Timeline

View the timeline of events for the virtual machines registered to a provider.

1. Navigate to **Compute > Infrastructure > Providers**.
2. Click a provider.
3. Click (**Monitoring**), and then (**Timelines**)from the taskbar, or from the provider accordion, click **Properties > Timeline**.
4. From **Options**, customize the period of time to display and the types of events to see.



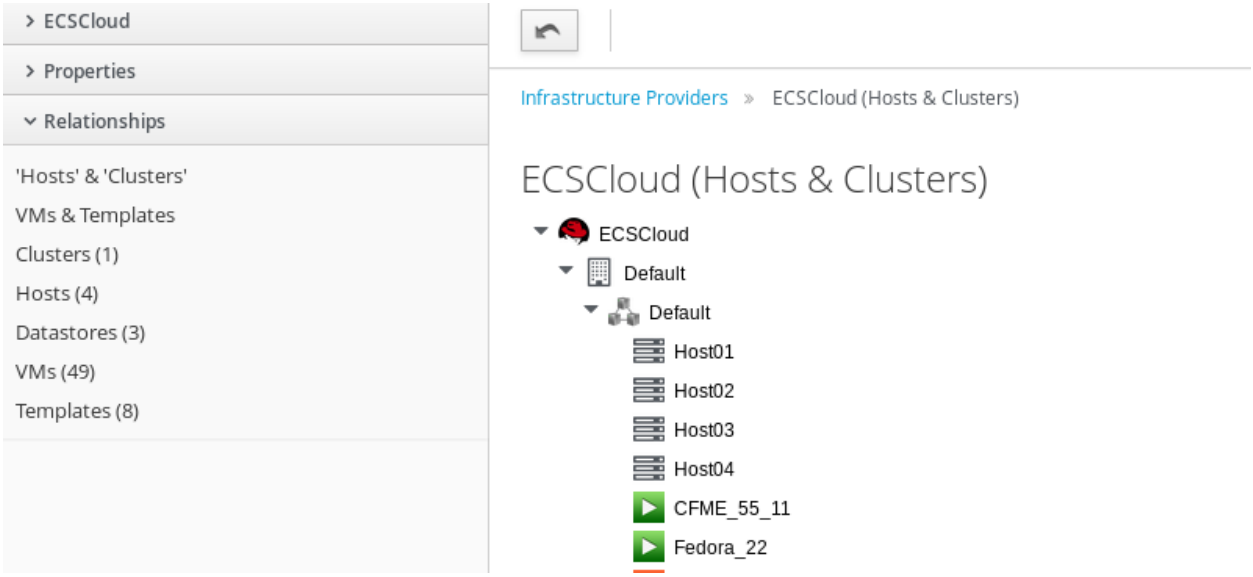
- Use **Show** to select regular Management Events or Policy Events.
- Use the **Interval** dropdown to select hourly or daily data points.

- Use **Date** to type the date for the timeline to display.
- If you select to view a daily timeline, use **Show** to set how many days back to go. The maximum history is 31 days.
- The three **Event Groups** lists allow you to select different groups of events to display. Each has its own color.
- From the **Level** list, select a **Summary** event, or a **Detail** list of events. For example, the detail level of a **Power On** event might include the power on request, the starting event, and the actual **Power On** event. If you select **Summary**, only the Power On event displays in the timeline.

Viewing Hosts and Clusters

Access a tree view of the hosts and clusters for a provider from the **Provider Summary**.

1. Navigate to **Compute > Infrastructure > Providers**.
2. Click the provider to view the hosts and clusters.
3. Click on the **Relationships** accordion, then click **Hosts & Clusters**.



Viewing Virtual Machines and Templates

Access a tree view of the virtual machines and templates for a provider from the **Provider Summary**.

1. Navigate to **Compute > Infrastructure > Providers**.
2. Click the provider to view the virtual machines and templates.
3. From accordion menu, click **Relationships**, then click **VMs & Templates**.

Configuration Management Providers

In Infrastructure Automation, a configuration management provider is a systems management product that you can add to a Infrastructure Automation appliance to manage the lifecycle of your resources. Configuration management providers are useful for uniformly applying changes and updates across providers, and for recording and reporting status and change activity. They can also help eliminate the confusion and error brought about by the existence of different providers.

This chapter describes the different types of configuration management providers available to Infrastructure Automation, and how to manage them. Configuration management providers must be added individually to Infrastructure Automation.

- IBM Terraform provider

- Red Hat Satellite 6

Automation Management Providers

In Infrastructure Automation, an automation management provider is a management tool that integrates with Infrastructure Automation to simplify automation operations for your resources. The following chapter describes the automation management providers that you can use with Infrastructure Automation, and how to work with them.

Infrastructure Automation provides automation management through the following features:

Automate enables real-time, bidirectional process integration. This embedded automate feature provides you with a method to implement adaptive automation for management events and administrative or operational activities.

Embedded Workflows provide the capability to simplify service catalog creation and improve the overall service catalog experience through a step-based automation system. Workflows can run docker containers to accomplish a task that the user requires as part of the state machine. Workflows are imported from Git repository content into Infrastructure Automation where they are made available for users to run. *Embedded Workflows* are an alternative to the *Embedded Automate* capability for authoring and running automation. With *Embedded Workflows*, you can continue to use the *Embedded Automate* feature for service catalog creation and management.

Ansible integration delivers out-of-the-box support for backing service, alert, and policy actions by using Ansible playbooks. Sync your existing playbook repositories with Infrastructure Automation, add credentials to access providers, and create service catalog items for actions ranging from creating and retiring VMs, updating security software, or adding more disks when space runs low.

Ansible Tower is a management tool that is integrated with Infrastructure Automation, designed to help automate infrastructure operations utlizizing existing Ansible Tower providers in your inventory. Infrastructure Automation allows you to execute Ansible Tower jobs by using service catalogs and Automate. Using Ansible Tower, you can schedule Ansible playbook runs and monitor current and historical results, allowing for troubleshooting or identification of issues before they occur.

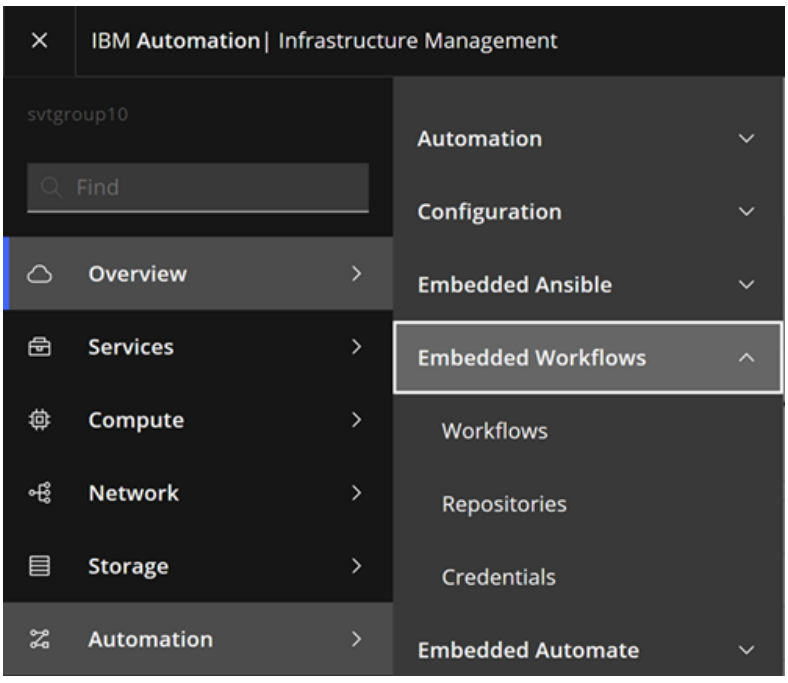
Embedded Workflows

Embedded Workflows provide the capability to simplify service catalog creation and improve the overall service catalog experience through a step-based automation system. Workflows can run docker containers to accomplish a task that the user requires as part of the state machine. Workflows are imported from Git repository content into Infrastructure Automation where they are made available for users to run.

Embedded Workflows are an alternative to the *Embedded Automate* capability for authoring and running automation. With *Embedded Workflows*, you can continue to use the *Embedded Automate* feature for service catalog creation and management.

Embedded Workflows UI

For the *Embedded Workflows* feature, the following pages are available in the UI, which you can use to access and work with workflows. To view these pages, go to **Automation** > **Embedded Workflows** and select the page.



- **Workflows**

  This UI page includes a summary of the available workflows. You can use different views to review the workflows:

  - Text view, which is the raw representation (JSON or YAML) of the Amazon State Languages (ASL) code.
  - Graph view, which shows a visual representation of the workflow structure.
- **Repositories**

  This page lists, and provides a summary of, the added embedded workflow repositories. From this page, you can add, update, and remove repositories. When a repository is added and synchronized, any workflows that are found in the repository can be viewed on the **Workflows** UI page.

- **Credentials**

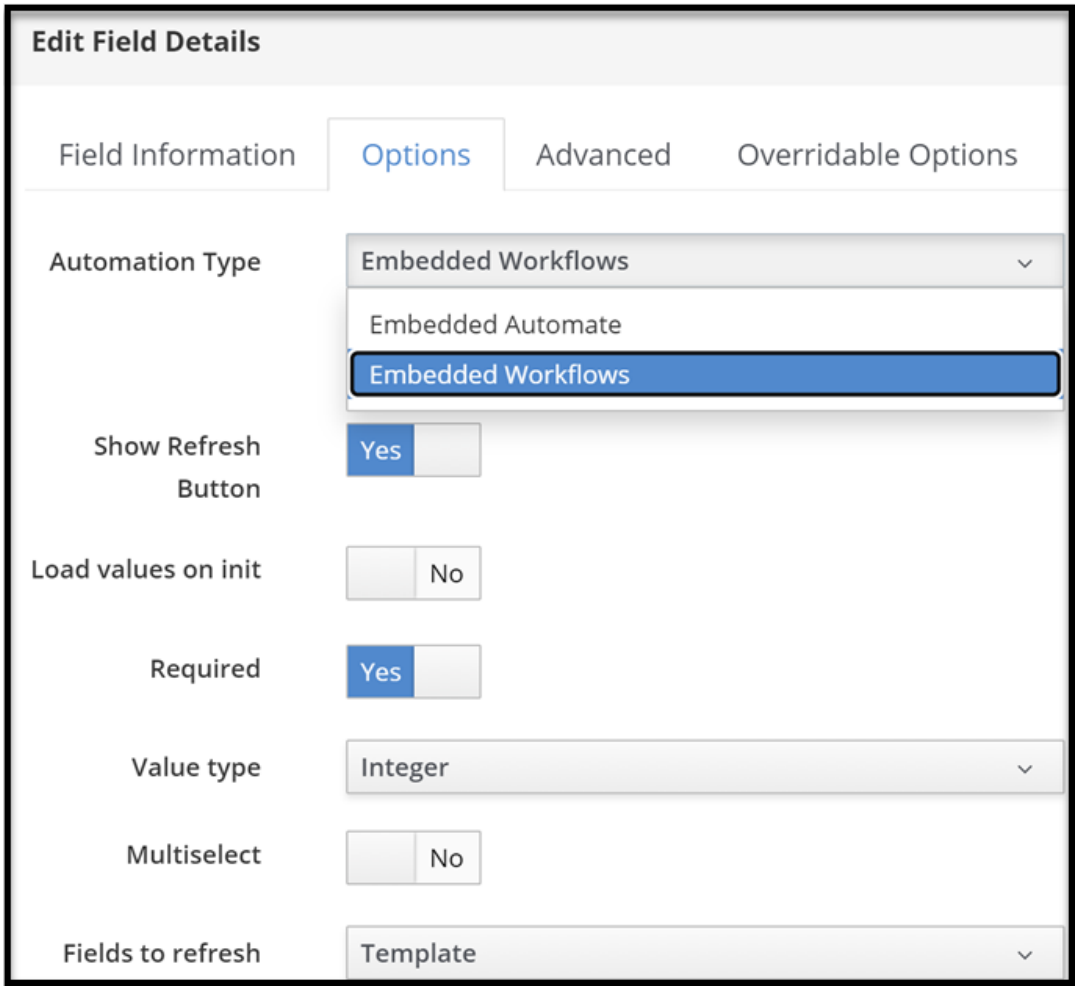
  Credentials are used by Infrastructure Automation for any authentication that is required when running embedded workflows, and when importing workflows from a version control system.

  This page lists, and provides a summary of, the credentials for accessing and using embedded workflows. From this page, you can add, update, and delete credentials.

In addition to the *Embedded Workflows* UI, **Embedded Workflows** can also be used in Service Dialogs for dynamic dialog fields and with Service Catalog Items for provisioning, reconfiguration, and retirement entrypoints.

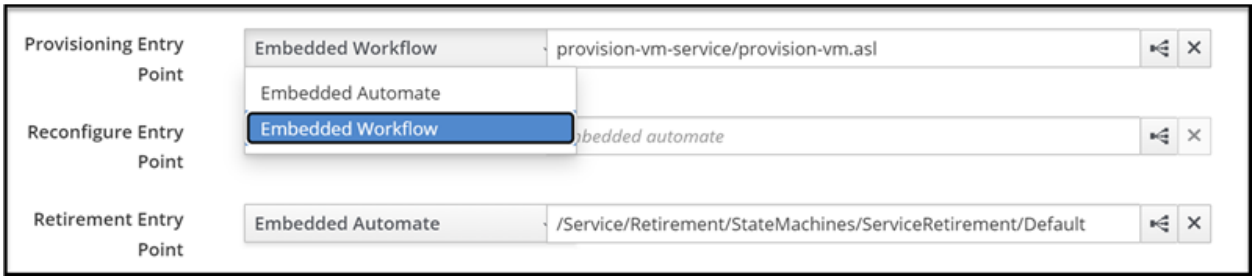
- Service Dialog

  When creating a service dialog, fields that are dynamic include the option for using **Embedded Workflows** or **Embedded Automate**.



- Service Catalog Items

  The provisioning, reconfigure, and retirement entry points include the option for using **Embedded Workflows** or **Embedded Automate**. You can set the point field to the workflow that you want to use for any, or all, of the three service entry points.

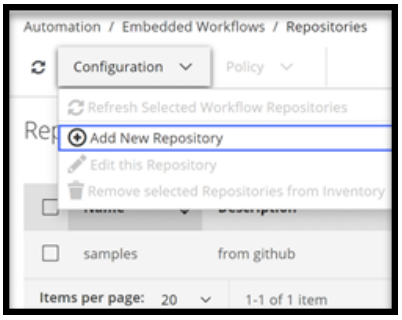


Adding an Embedded Workflows Repository

As a starting point to begin using workflows, you must first add a repository that includes your workflows.

To add a repository, complete the following steps:

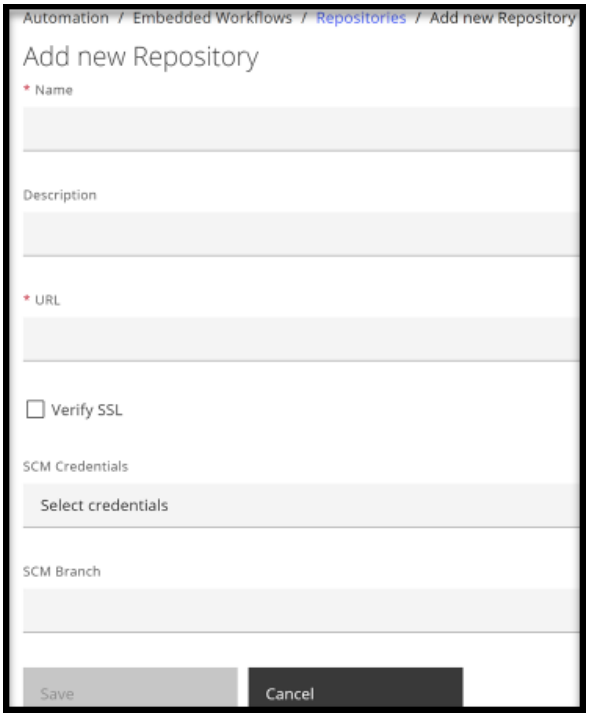
1. Click **Automation** > **Embedded Workflows** > **Repositories** > **Configuration** > **Add New Repository**.



2. Enter the repository name in the **Name** field.
3. Add a description for the repository in the **Description** field.
4. Enter a Git repository URL for the repository in the **URL** field. The Git repository URL does not need to be a GitHub repository.
5. If you are adding a repository that requires authentication then add an SCM (Source Control Management) credential for accessing the repository. Select that credential for this repository from the **SCM Credentials** dropdown menu. These credentials are used in different ways within embedded workflows.

   Source Control credentials have several attributes that can be configured:

   - **Username**: The username to use in conjunction with the source control system.
   - **Password**: The password to use in conjunction with the source control system.
   - **Private key passphrase**: If the SSH private key used is protected by a passphrase, you can configure a key passphrase for the private key.
   - **Private Key**: Copy or drag and drop the actual SSH Private Key to be used to authenticate the user to the source control system by using SSH.
6. Provide a branch name in the **SCM Branch** field for the branch that you want to pull.
7. Click **Save**. Upon saving, the repository is imported and scanned for any included workflows (any Amazon States Language `.asl` files).



8. Confirm that the synchronization with your repository is successful by viewing the list of workflows that are detected for the repository.

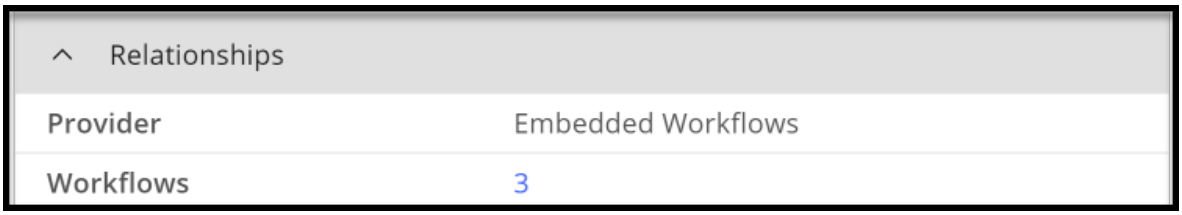
To view the added repositories, return to the **Automation** > **Embedded Workflows** > **Repositories** page. The list of repositories is displayed. To view a summary of details for a repository, click the repository to open the *Summary* page for that repository.

If you need to edit a repository, select the repository from the list of repositories. Then, click **Configuration** > **Edit this Repository**. After you complete your edits, click **Save**.

If you need to delete a repository, select the repository from the list of repositories. Then, click **Configuration** > **Remove this Repository from Inventory**.

Viewing the List of Workflows in a Repository

When your repository is added, the repository shows in the *Repositories* list. You can now click the entry for the repository to open it so that you can view the imported workflows. If any workflows are found, click the corresponding number to view the list of workflows.



Alternatively, click **Automation** > **Embedded Workflows**. Then, click **Workflows** to see the list of workflows across all repositories.

Refreshing Repositories

Infrastructure Automation allows you to refresh a targeted repository or all repositories in your inventory to ensure that your workflows are current.

Refresh a targeted repository:

1. Browse to the menu: **Automation** > **Embedded Workflows** > **Repositories**.
2. Click a repository.
3. Click **Configuration**, then (**Refresh this Repository**).

Alternately, you can refresh some or all repositories from the list view:

1. Browse to the menu: **Automation** > **Embedded Workflows** > **Repositories**.
2. Check those repositories to refresh. Click **Check All** to select all repositories.

3. Click **Configuration**, then (**Refresh Selected Workflow Repositories**).

Adding Credentials

Credentials are used by Infrastructure Automation for any authentication that is required when running embedded workflows, or when importing workflows from a version control system.

Infrastructure Automation can store credentials that are used by embedded workflows. Credentials that are saved in Infrastructure Automation are mapped to values in the `Credentials` field of your Workflow and are made available to the container with a workflow when the workflow runs.

1. Browse to the menu: **Automation** > **Embedded Workflows** > **Credentials**.
2. Click **Configuration**, then (**Add New Credential**).
3. Provide a **Name** for the credential.
4. Select the **Credential Type**. More fields appear depending on the type chosen.
5. Click **Add**.

To view the added credentials, return to the **Automation** > **Embedded Workflows** > **Credentials** page. The list of credentials is displayed. To view a summary of details for a credential, click the credential to open the *Summary* page for that credential.

If you need to edit a credential, select the credential from the list of credentials. Then, click **Configuration** > **Edit this Credential**. After you complete your edits, click **Save**.

If you need to delete a credential, select the credential from the list of credentials. Then, click **Configuration** > **Remove selected Credentials from Inventory**.

Credential Types

Each credential type that is used by Infrastructure Automation for embedded workflows is detailed in the following sections.

SCM

SCM (source control Management) credentials are used with Projects to clone and update local source code Git repositories. Source Control credentials have several attributes that can be configured:

- **Username**: The username to use in conjunction with the source control system.
- **Password**: The password to use in conjunction with the source control system.
- **Private key passphrase**: If the SSH private key used is protected by a passphrase, you can configure a key passphrase for the private key.
- **Private Key**: Copy or drag-and-drop the actual SSH Private Key to be used to authenticate the user to the source control system by using SSH.

Authoring a workflow

You can configure workflows for use in the following areas during service catalog creation:

- **Dynamic dialog fields** - You (or your users) can attach workflows to a field so that the field becomes populated or refreshed with the results from the running of the workflow.
- **Service Entry Points** - You (or your users) can set and use workflows for provisioning, reconfiguring, and retiring services.

You can create and use embedded workflows as needed to not only change parts of the provisioning process, but also to automate other operational tasks.

Workflows must be authored in Amazon State Languages (ASL) format. As part of authoring a workflow, you (or your users) can build container images that are able to perform any tasks that are required in any language that you like. You can use these images during Task states in your workflows.

1. Define the code for the workflow. If your workflow requires the use of any credentials or parameters to be specified, ensure that they are passed in the code.

   Within the workflow code, you need to specify the states that your workflow requires, including any next steps. For `Task` type steps in the workflow, a docker container is called. The container defines what happens for that Task state. For example, a docker container can run to clone a template. If your states require parameters or credentials, you can specify them in your state definitions.

   The workflow code must be in the Amazon States Language (ASL) format and follow its supported specifications. For more information about Amazon States Language and its specification, see Amazon States Language Guide.

2. Build the docker containers that are required for the workflow.

   When you have the code for your task resource written, you need to bundle it into a docker image. You can bundle the code by creating a standard Dockerfile and building the image (https://docs.docker.com/engine/reference/commandline/build/). Then, you can push the image to a registry, which makes the image available to be used by Infrastructure Automation. When you have pushed your images to an image registry, you can add the registry to Infrastructure Automation.

Pull secrets for containers are used differently between appliances and the OpenShift Container Platform (OCP). These differences are outlined in the following sections.

Running an Embedded Workflow on Appliances

- On appliances, **podman** is used to execute the container so use podman login as the **manageiq** user.

  ```
  # su manageiq
  $ podman login docker.io
  Username:
  Password:
  Login Succeeded!
  ```

  If you use the –root flag in the podman pull, images are pulled to a local directory '/var/www/miq/vmdb/data/containers/storage' as in the example

  ```
     podman pull <repository>/<image>:<tag>  --root /var/www/miq/vmdb/data/containers/storage
  ```

  It is worth noting that the default /home/manageiq partition has insufficient space to store large images.

  You can use any repository to store your images, for example you can use docker.io access token so that the token does not expire.

- Provide an image pull secret to a podified Kubernetes container, and then add it to a service account

  In order to pull an image from a private registry you have to provide an **ImagePullSecret** to your containers, see Pull an Image from a Private Registry. Infrastructure Automation uses a service account called ibm-infra-management-application-default to run containers for your workflows. You can add an **ImagePullSecret** to this service account by following Add Image Pull Secrets to a service account.

Running an Embedded Workflow on the OpenShift Container Platform (OCP)

If the user is running an embedded workflow on OCP, and is using a docker repository requiring a custom pull secret for their container images, the following steps need to be carried out.

1. Create a pull secret for the repository in question in the same namespace as Infrastructure Automation, for example:

   ```
    oc create secret docker-registry <pull-secret-name> --docker-server=<docker-server> --docker-username=
   <docker-username> --docker-password=<docker-password>
   ```

2. Add this pull secret to the ibm-infra-management-application-default, for example:

   ```
    oc secret link ibm-infra-management-application-default <pull-secret-name> --for=pull
   ```

Example: Provisioning Workflow

```
{
  "Comment": "Provision a VMware VM.",
  "StartAt": "CloneTemplate",
  "States": {
    "CloneTemplate": {
      "Type": "Task",
      "Resource": "docker://docker.io/manageiq/workflows-examples-clone-template:latest",
      "Next": "CheckTaskComplete",
      "Credentials": {
        "api_user.$": "$.api_user",
        "api_password.$": "$.api_password",
        "vcenter_user.$": "$.vcenter_user",
        "vcenter_password.$": "$.vcenter_password"
      },
      "Parameters": {
        "PROVIDER_ID.$": "$.dialog_provider",
        "VERIFY_SSL": false,
        "TEMPLATE.$": "$.dialog_source_template",
        "NAME.$": "$.dialog_vm_name"
      }
    },

    "CheckTaskComplete": {
      "Type": "Task",
      "Resource": "docker://docker.io/manageiq/workflows-examples-check-task-complete:latest",
      "Next": "PollTaskComplete",
```

```
      "Credentials": {
        "vcenter_user.$": "$.vcenter_user",
        "vcenter_password.$": "$.vcenter_password"
      },
      "Parameters": {
        "VCENTER_HOST.$": "$.vcenter_host",
        "TASK.$": "$.task"
      }
    },

    "PollTaskComplete": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.state",
          "StringEquals": "success",
          "Next": "PowerOnVM"
        },
        {
          "Variable": "$.state",
          "StringEquals": "running",
          "Next": "RetryState"
        },
        {
          "Variable": "$.state",
          "StringEquals": "error",
          "Next": "FailState"
        }
      ],
      "Default": "FailState"
    },

    "RetryState": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "CheckTaskComplete"
    },

    "PowerOnVM": {
      "Type": "Task",
      "Resource": "docker://docker.io/manageiq/workflows-examples-power-on-vm:latest",
      "Next": "SuccessState",
      "Credentials": {
        "vcenter_user.$": "$.vcenter_user",
        "vcenter_password.$": "$.vcenter_password"
      },
      "Parameters": {
        "VCENTER_HOST.$": "$.vcenter_host",
        "VM.$": "$.vm"
      }
    },

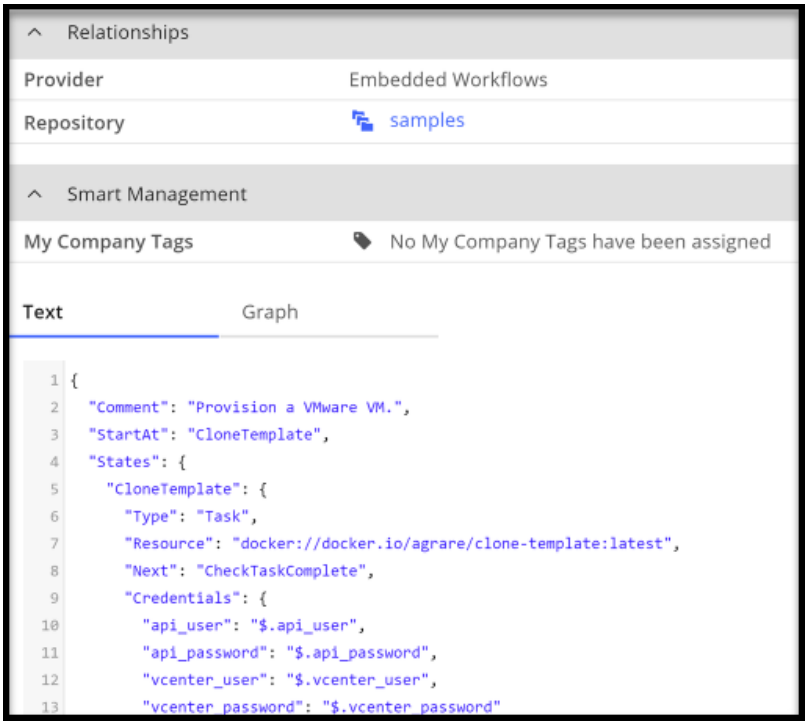
    "FailState": {
      "Type": "Fail",
      "Error": "FailStateError",
      "Cause": "No Matches!"
    },

    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
```

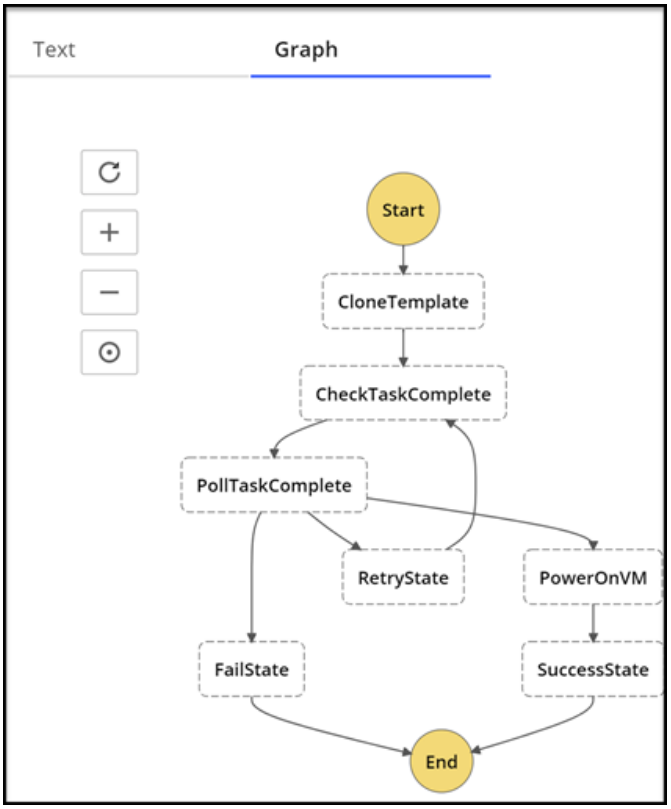
Viewing workflow details

Once a workflow is imported, you can view the details, such as to verify the payload. To view the details for a workflow:

1. Go to the menu: **Automation** > **Embedded Workflows**. Then, click **Workflows** to see the workflows list.
2. Click a workflow. The default *Text* view of the workflow opens where you see the code content.



Alternatively, click the **Graph** tab to see a graph of the same data.



3. If your workflow has credential fields, create a workflow credential for each of the fields. For more information about creating a credential, see Adding Credentials.
4. Map any workflow credentials to the workflow.
    1. Go to the menu: **Automation** > **Embedded Workflows**. Then, click **Workflows** to see the workflows list.
    2. Select the workflow that includes the credentials that you need to map.
    3. Click **Configuration** > **Map Credentials to this Workflow**.

The table that displays on the *Map Credentials* page displays all the current mappings for the selected workflow. Verify that the list of credential identifiers includes all of the identifiers that are in the workflow payload.

4. Use the drop-down fields to map one or more credentials for the workflow. Verify that the table is updated with the correct information.
5. Click **Save**.

Creating a Service Dialog using an Embedded Workflow

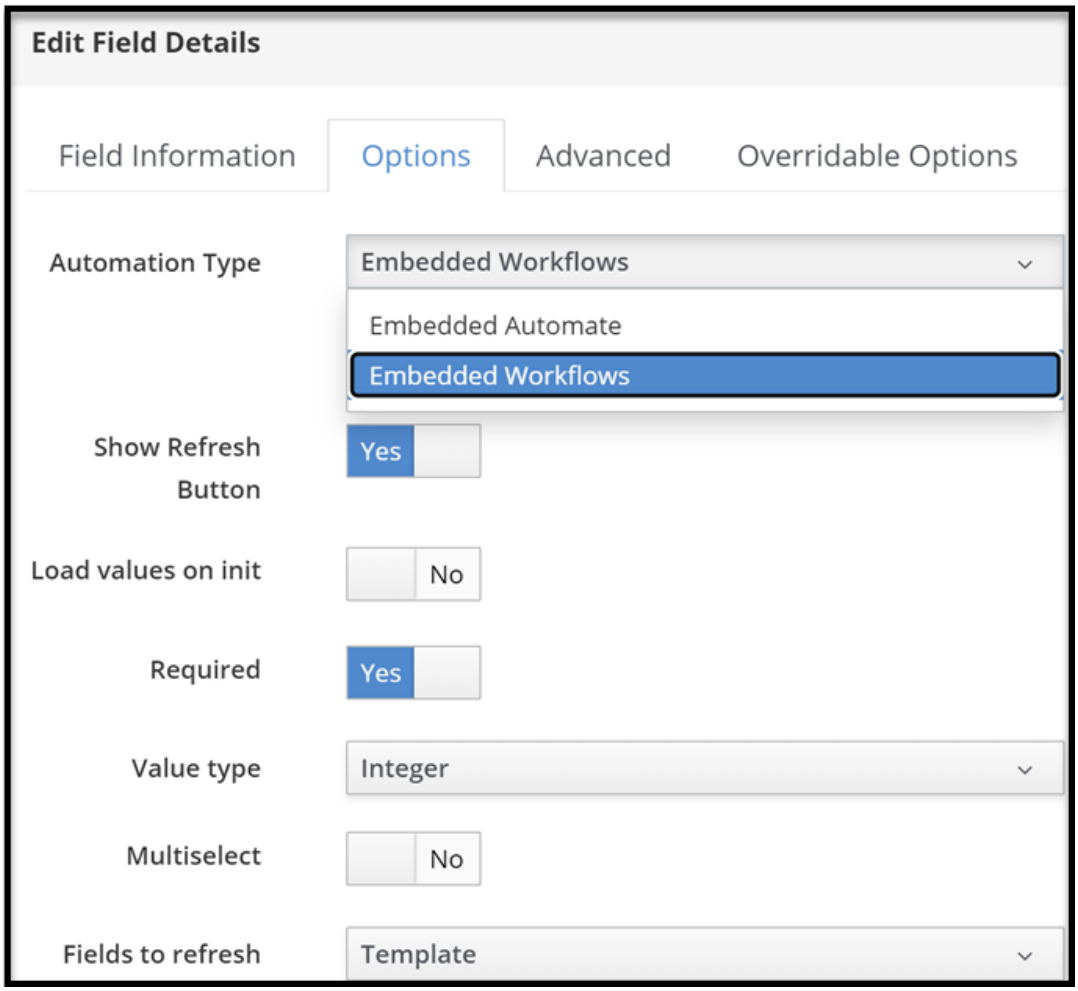
Fields that are *Dynamic* offer the choice of *Embedded Workflows* in addition to the original *Embedded Automate*.

Dialogs are the part of Infrastructure Automation that you interface with when you order services.

1. To open a service dialog, click **Automation** > **Embedded Automate** > **Customization**.
2. From the *Service Dialog* section, review the list of created dialogs. Click one of the dialogs to view the details.
3. Click **Edit the dialog**. You can edit the *Name*, *Provider*, and *Template*.

   When you select to edit any of the dropdown settings, you can see a *dynamic* option. Toggle to **On** to make sure it is enabled.

4. Click the **Options** tab. You can now select between *Embedded Automate* and *Embedded Workflows* in the Automation field.

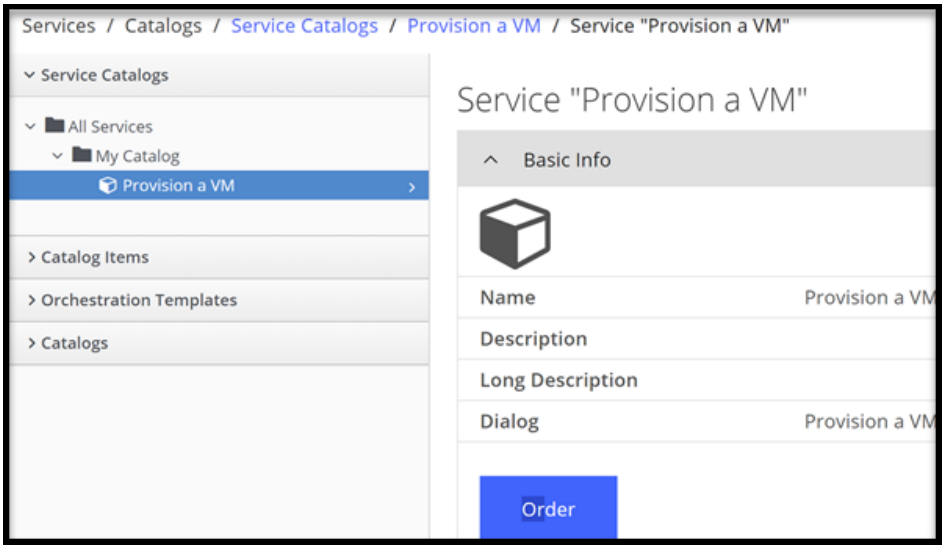


If you choose **Embedded Workflows**, you can select the workflow to run. Complete the remaining fields and click **Save** when you complete the form. You can chain dialog fields together by using **Fields to refresh**, which lets one dialog field trigger the run of another when there are interdependencies. When you *order* a service catalog, any dynamic dialogs that are backed by embedded workflows run in the background to populate the dialogs.

Creating Service Catalog Items using an Embedded Workflow

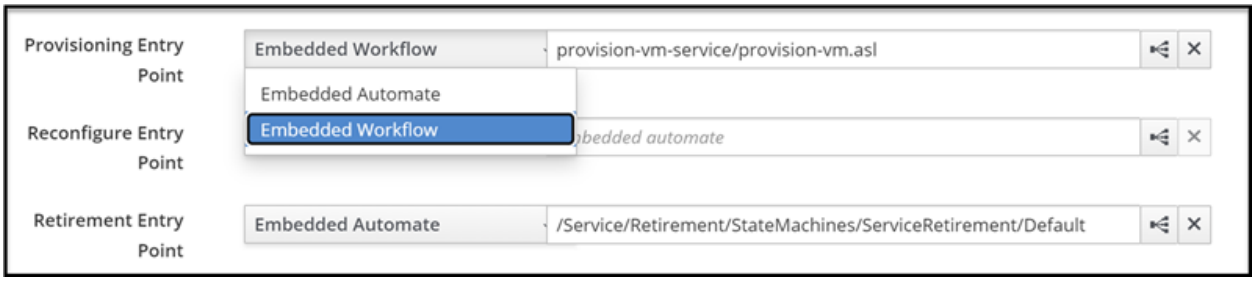
You can create a generic service catalog item that uses an embedded workflow. To use previously set up workflows and to view a service catalog item, you need to *order* the item. Complete the following steps to order a service catalog item that uses embedded workflows:

1. Click **Services** > **Catalogs**.
2. The *Service Catalogs* section opens and you see the existing Service Catalogs. If you want to order one, click it and then **Order**.



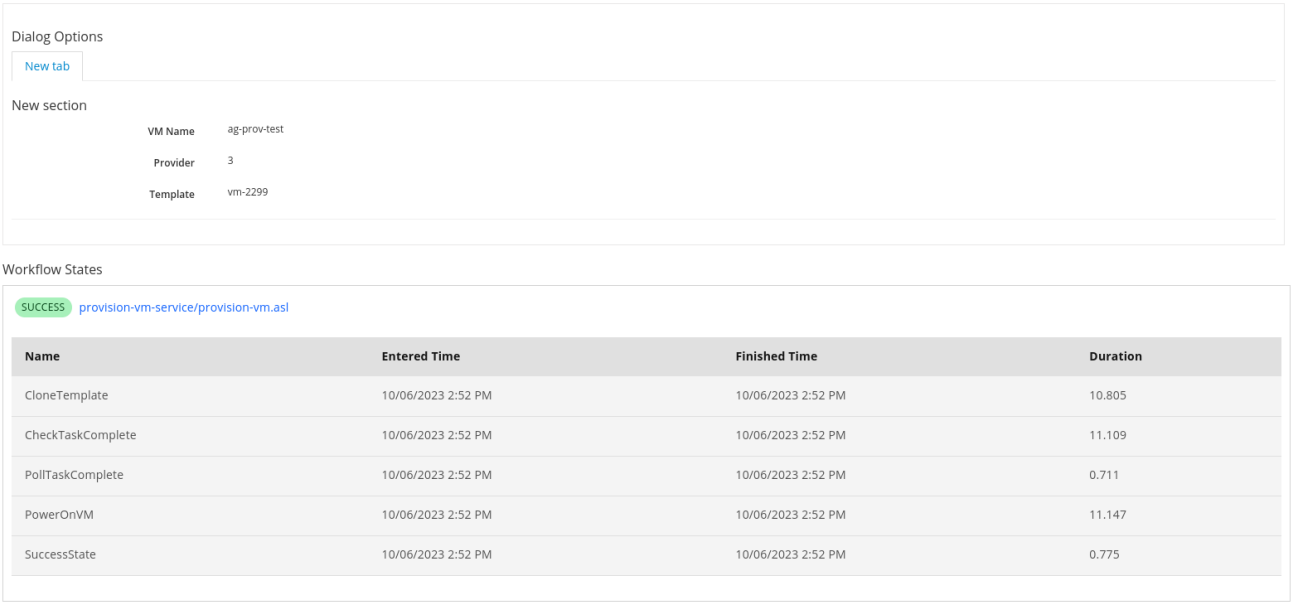
Complete the fields for your specified dialog. Then, click **Submit**.

3. If you want to know more about the item first or to edit it, click the **Catalogs** section, and then click that item that is created. Click **Configuration** > **Edit this item**.



4. When you complete all your edits, click **Save**.
5. Confirm that you can order the service catalog item and that it runs the workflows to drive the dynamic dialog dropdown.

   The list of services and requests is shown when the catalog item is submitted. Clicking the request shows the execution status, including any embedded workflows.



Ansible

Ansible integrates with Infrastructure Automation to provide automation solutions, using playbooks, for Service, Policy and Alert actions. Ansible playbooks consist of series of *plays* or tasks that define automation across a set of hosts, which is known as the inventory.

Ranging from simple to complex tasks, Ansible playbooks can support cloud management:

- **Services** - allow a playbook to back a Infrastructure Automation service catalog item.
- **Control Actions** - Infrastructure Automation policies can execute playbooks as actions based on events from providers.
- **Control Alerts** - set a playbook to launch prompted by a Infrastructure Automation alert.

Ansible is built into Infrastructure Automation so there is nothing to install. The basic workflow when using Ansible in Infrastructure Automation is as follows:

1. Enable the **Embedded Ansible** server role.
2. Add a source control repository that contains your playbooks.
3. Establish credentials with your inventory.
4. Back your services, alerts, and policies using available playbooks.

Enabling the Embedded Ansible Server Role

In Infrastructure Automation, the **Embedded Ansible** role is disabled by default. Enable this server role to utilize Ansible Automation Inside.

Note: Configure your Infrastructure Automation appliance network identity (hostname/IP address) before enabling the Embedded Ansible server role. Restart the **`evmserverd`** service on the appliance with the enabled Embedded Ansible server role after making any changes to the hostname or IP address.

1. Browse to the settings menu, then **Configuration > Settings**.
2. Select the desired server under **Zones**.
3. Set the **Server Role** for **Embedded Ansible** to **On**.

Verifying the Embedded Ansible Worker State

Verify that the Embedded Ansible worker has started to utilize its features.

1. Browse to the settings menu, then **Configuration > Diagnostics** and click on the desired server.
2. Click on the **Roles by Servers** tab.

A table of all workers and current status will appear from which you can confirm the state of your embedded Ansible worker.

Adding a Playbook Repository

Add a repository so that Infrastructure Automation can discover and make available your playbooks.

1. Browse to menu: **Automation > Ansible > Repositories**.
2. Click **Configuration**, then (**Add New Repository**).
3. Provide a Repository Name in the **Name** field.
4. Add a description for the repository in the **Description** field.
5. Add a **URL** or IP Address for the repository.
6. Select the appropriate **SCM Credentials** from the drop-down menu.
7. Provide a branch name in the **SCM Branch** field.
8. Click **Save**.

Once you have synced a repository, its playbooks will become available to Infrastructure Automation.

Refreshing Repositories

Infrastructure Automation allows you to refresh a targeted playbook repository or all repositories in your inventory to ensure your playbooks are current.

Refresh a targeted repository:

1. Browse to menu: **Automation > Ansible > Repositories**.
2. Click on a repository.

3. Click **Configuration**, then (**Refresh this Repository**).

Alternately, you can refresh some or all repositories from the list view:

1. Browse to menu: **Automation > Ansible > Repositories**.
2. Check those repositories to refresh. Click **Check All** to select all repositories.
3. Click **Configuration**, then (**Refresh Selected Ansible Repositories**).

Credentials

Credentials are utilized by Infrastructure Automation for authentication when running Ansible playbooks against machines, synchronizing with inventory sources, and importing project content from a version control system.

Adding Credentials

Infrastructure Automation can store credentials used by playbooks. Credentials saved in Infrastructure Automation are matched and executed with a playbook when run.

1. Browse to menu: **Automation > Ansible > Credentials**.
2. Click **Configuration**, then (**Add New Credential**).
3. Provide a **Name** for the credential.
4. Select the **Credential Type**. Additional fields will appear depending on the type chosen.
    - **Vault Password**: Ansible Vault credentials have only the **Vault Password** attribute that may be configured. For more information on Ansible Vault, see Using Vault in playbooks.
5. Click **Add**.

Credential Types

Each credential type used by Infrastructure Automation is detailed in the following sections.

Machine

Machine credentials enable Infrastructure Automation to invoke Ansible on hosts under your management. Just like using Ansible on the command line, you can specify the SSH username, optionally provide a password, an SSH key, or a key password. They define SSH and user-level privilege escalation access for playbooks, and are used when running playbooks on a remote host.

- **Username**: The username to be used for SSH authentication.
- **Password**: The actual password to be used for SSH authentication.
- **SSH Private Key**: Copy or drag-and-drop the SSH private key for the machine credential.
- **Private Key Passphrase**: If the SSH Private Key used is protected by a password, you can configure a Key Password for the private key.
- **Privilege Escalation**: Specifies the type of escalation privilege to assign to specific users. Options include **sudo**, **su**, **pbrun**, **pfexec**.
- **Privilege Escalation Username**: Enter the username to use with escalation privileges on the remote system.
- **Privilege Escalation Password**: Enter the actual password to be used to authenticate the user via the selected privilege escalation type on the remote system.

Network

Network credentials are used by Ansible networking modules to connect to and manage networking devices.

Network credentials have several attributes that may be configured:

- **Username**: The username to use in conjunction with the network device.
- **Password**: The password to use in conjunction with the network device.
- **Authorize**: Select this from the Options field to add an Authorize password which signs the RSA key with a password.

- **Authorize password**: If **Authorize** is checked, enter a password in the **Authorize Password** field.
- **SSH Key**: Copy or drag-and-drop the actual SSH Private Key to be used to authenticate the user to the network via SSH.
- **Private key passphrase**: The actual passphrase for the private key to be used to authenticate the user to the network via SSH.

SCM

SCM (source control) credentials are used with Projects to clone and update local source code repositories from a remote revision control system such as Git, Subversion, or Mercurial.

Source Control credentials have several attributes that may be configured:

- **Username**: The username to use in conjunction with the source control system.
- **Password**: The password to use in conjunction with the source control system.
- **Private key passphrase**: If the SSH private key used is protected by a passphrase, you may configure a key passphrase for the private key.
- **Private Key**: Copy or drag-and-drop the actual SSH Private Key to be used to authenticate the user to the source control system via SSH.

Amazon

Selecting this credential type enables synchronization of cloud inventory with Amazon Web Services.

- **Access Key**: User credentials that allow for programmatic calls to Amazon Web Services.
- **Secret Key**: The secret key that corresponds to the user access key.
- **STS Token**: Token generated by Amazon Web Services Security Token Service.

Azure

Selecting this credential type enables synchronization of cloud inventory with Microsoft Azure.

Microsoft Azure credentials have several attributes to configure:

- **Username**: The username to use to connect to the Microsoft Azure account.
- **Password**: The password to use to connect to the Microsoft Azure account.
- **Subscription ID**: The Subscription UUID for the Microsoft Azure account.
- **Tenant ID**: The Tenant ID for the Microsoft Azure account.
- **Client Secret**: The Client Secret for the Microsoft Azure account.
- **Client ID**: The Client ID for the Microsoft Azure account.

OpenStack

Selecting this credential type enables synchronization of cloud inventory with Red Hat OpenStack Platform.

OpenStack credentials have several attributes that may be configured:

- **Username**: The username to use to connect to OpenStack.
- **Password (API Key)**: The password or API key to use to connect to OpenStack.
- **Host (Authentication URL)**: The host to be used for authentication.
- **Project (Tenant Name)**: The Tenant name or Tenant ID used for OpenStack. This value is usually the same as the username.
- **Domain name**: The FQDN to be used to connect to OpenStack.

Red Hat Virtualization

Selecting this credential type enables synchronization of cloud inventory with Red Hat Virtualization.

Red Hat Virtualization credentials have several attributes that may be configured:

- **Username**: The username to use to connect to Red Hat Virtualization.
- **Password**: The password to use to connect to Red Hat Virtualization.
- **Host (Authentication URL)**: The host to be used for authentication.

Important: Enter in **Host** the Red Hat Virtualization provider URL, followed by the path **`/ovirt_engine/api`**. Example: **`https://your.rhv.com/ovirt_engine/api`**

For more information on the Ansible Roles that are available for Red Hat Virtualization, see Ansible Roles.

VMware

Selecting this credential type enables synchronization of inventory with VMware vCenter.

Important: If both Infrastructure Automation and a VMware provider are located in the same IPv6-only network, use a DNS-resolvable hostname for the VMware provider in the **vCenter Host** field when adding credentials.

VMware credentials have several attributes that may be configured:

- **Username**: The username to use to connect to vCenter.
- **Password**: The password to use to connect to vCenter.
- **vCenter Host**: The vCenter hostname or IP address to connect to.

Note: If the VMware guest tools are not running on the instance, VMware inventory sync may not return an IP address for that instance.

Tagging Ansible Playbooks, Repositories, and Credentials

Apply tags to Ansible playbooks, repositories, and credentials to categorize them. Tagging enables administrators to limit users to view those Ansible elements that have been enabled for that set of user permissions.

Adding Tags to Ansible Playbooks

1. Navigate to **Automate > Ansible > Playbooks**.
2. Select the checkboxes for the Ansible playbooks to tag.
3. Click **Policy**, and then (**Edit Tags**).
4. Select a customer tag to assign from the first list.
5. Select a value to assign from the value list.
6. Click **Save**.

Adding Tags to Ansible Repositories

1. Navigate to **Automate > Ansible > Repositories**.
2. Select the checkboxes for the Ansible repositories to tag.
3. Click **Policy**, and then (**Edit Tags**).
4. Select a customer tag to assign from the first list.
5. Select a value to assign from the value list.
6. Click **Save**.

Adding Tags to Ansible Credentials

1. Navigate to **Automate > Ansible > credentials**.
2. Select the checkboxes for the Ansible credentials to tag.
3. Click **Policy**, and then (**Edit Tags**).
4. Select a customer tag to assign from the first list.
5. Select a value to assign from the value list.
6. Click **Save**.

Optimizing Ansible Playbooks for Infrastructure Automation

Ansible is a simple model-driven configuration management, multi-node deployment, and remote-task execution system. When designing playbooks for use with Infrastructure Automation it is helpful to utilize solutions within the playbook itself to ensure optimal implementation of playbook-backed services or automated processes.

This section is intended to complement the existing documentation on Ansible playbooks and guide administrators through optimizing playbooks for use with Infrastructure Automation.

Installing Roles on an Embedded Ansible Appliance

Roles are ways of automatically loading certain variable files, tasks, and handlers based on a known file structure. Grouping content by roles also allows for easy sharing of roles with other users. Install roles on a Infrastructure Automation appliance with the Embedded Ansible server role activated to optimize playbooks.

When using this role in a playbook on a Infrastructure Automation appliance, add an empty **`roles`** directory at the root of the playbook. In the **`roles`** directory, include a **`requirements.yml`** file with the following contents:

```
---
- src: <ansible-galaxy-role>
```

Infrastructure Automation automatically installs the role once it sees the **`requirements.yml`** file in the playbook.

Ansible Service Linking

Infrastructure Automation provides a module allowing inventoried resources such as virtual machines created using Ansible playbooks to link back to the services used to generate them. During service ordering of a playbook the **`add_provider_vms`** module will allow the playbook to connect back to the worker appliance and identify the provider resources it was responsible for generating. Once linked, the newly generated resources are available to Infrastructure Automation's life cycle management features.

Linking VMs back to the service that created it requires implementing the following tasks in the playbook used for provisioning:

1. Create a resource and register it.
2. Link the service using the **`add_provider_vms`** method to the newly created resource.

Example: Linking a virtual machine to a service

In the following playbook task examples, a virtual machine is deployed to Amazon EC2 and linked back to the service. Examples are provided for linking the resource to its service by both an *href slug* and as an object.

Note:

- This example utilizes the `syncrou.manageiq-vmdb` role. This role allows Infrastructure Automation users to modify and/or change VMDB objects using an Ansible playbook. For information on implementing and utilizing roles when writing Ansible playbooks for Infrastructure Automation, see Installing Ansible Roles.
- For more information on Ansible Galaxy and roles, see the Ansible Galaxy documentation.
- Note the provider ID in order to successfully link to the service.

1. Create and register the resource.

```
- name: Create Ec2 Instance
  ec2:
    key_name: ""
    instance_tags: {Name: ""}
    group_id: ""
    instance_type: ""
    region: ""
    image: ""
    wait: yes
    count: 1
    vpc_subnet_id: ""
    assign_public_ip: yes
  register: ec2
```

2. Call the **`add_provider_vms`** method as an action to link to the service via an *href slug* or an object.

```
- name: Service Linking via an href slug
  manageiq_vmdb:
    href: "href_slug::services/80"
```

```
    action: add_provider_vms
    data:
      uid_ems:
        - ""
      provider:
        id: 24

- name: Service Linking via an object
  manageiq_vmdb:
    vmdb: ""
    action: add_provider_vms
    data:
      uid_ems:
        - ""
      provider:
        id: 24
```

Modifying the Automate Workspace Using the `manageiq-automate` Role.

The **manageiq-automate** role allows users of Infrastructure Automation Automate to modify and add to the automate workspace via an Ansible playbook.

Note: When using this role in a playbook on a Infrastructure Automation appliance with Embedded Ansible activated, add an empty **roles** directory at the root of the playbook. In the **roles** directory, include a **requirements.yml** file with the following contents:

```
---
- src: syncrou.manageiq-automate
```

Infrastructure Automation will automatically install the role once it sees the **requirements.yml** file in the playbook.

Role Variables

The **manageiq_automate** role employs the following variables when implemented in a playbook run on a Infrastructure Automation appliance. Variables are defined in **defaults/main.yml** and **vars/main.yml**.

auto_commit: By default is set to **True**. If set to False it does not auto commit back to Infrastructure Automation each call to a **set_** method in the **manageiq_automate** module.

manageiq_validate_certs: By default is set to **True**. If passed in with **extra_vars** or assigned in the playbook variables then the lookup allows self-signed certificates to be used when using SSL REST API connection URLs.

Example Playbook

The following example utilizes the **manageiq-automate** role. Using variable substitution, playbook tasks retrieve method parameters which are then used to modify object attributes. A final task uses the **set_retry** module to update the retry interval.

```
- name: Siphon Method Parameters into an object
  hosts: localhost
  connection: local
  vars:
  - auto_commit: True
  - object: root
  - interval: 600

  gather_facts: False
  roles:
  - syncrou.manageiq-automate

  tasks:
    - name: "Get the list of Method Parameters"
      manageiq_automate:
        workspace: ""
        get_method_parameters: yes
      register: method_params

    - name: "Set attributes"
      manageiq_automate:
        workspace: ""
        set_attributes:
          object: ""
          attributes: ""

    - name: Set Retry
      manageiq_automate:
        workspace: ""
        set_retry:
          interval: ""
```

Callbacks in Multiple Appliance Environments

In a Infrastructure Automation multiple appliance environment, enable the Embedded Ansible server role on a dedicated Infrastructure Automation appliance. Add `store_session:sql` to Ansible playbooks to ensure successful callbacks to Infrastructure Automation appliances in a multiple appliance environment.

Ansible Tower

Ansible Tower is a management tool integrated with Infrastructure Automation, designed to help automate infrastructure operations. Infrastructure Automation allows you to execute Ansible Tower jobs or workflows using service catalogs and Automate. No custom configuration or Ruby scripting is needed in Infrastructure Automation, as configuration is done in Ansible Tower using playbooks.

You can use the large library of existing Ansible playbooks as Infrastructure Automation state machines to automate tasks such as deployments, backups, package updates, and maintenance in your Infrastructure Automation environment. This can be particularly useful for quickly applying changes across large environments with many virtual machines or instances.

Using Ansible Tower, you can schedule Ansible playbook runs and monitor current and historical results, allowing for troubleshooting or identification of issues before they occur.

Infrastructure Automation supports Ansible Tower API v2 provider integration.

Working with an Ansible Tower Provider

The basic workflow when using Infrastructure Automation with an Ansible Tower provider is as follows:

1. Create an Ansible playbook which performs a specific task.
2. A new Ansible Tower job template is created from the playbook (or workflow template created from disparate jobs), which is then retrieved by Infrastructure Automation.
3. From the Ansible Tower job or workflow template, create a new catalog item in Infrastructure Automation, optionally with a service dialog that allows the user to enter parameters if needed.
4. The user orders the service from the Infrastructure Automation user interface, and fills out any additional arguments (for example, limiting the task to run on a specific set of virtual machines).
5. The job or workflow executes.

Notes:

- For more information about Ansible playbooks, see the Ansible playbook documentation.
- For more information about worklows, see Workflows in the Ansible Tower *User Guide*.

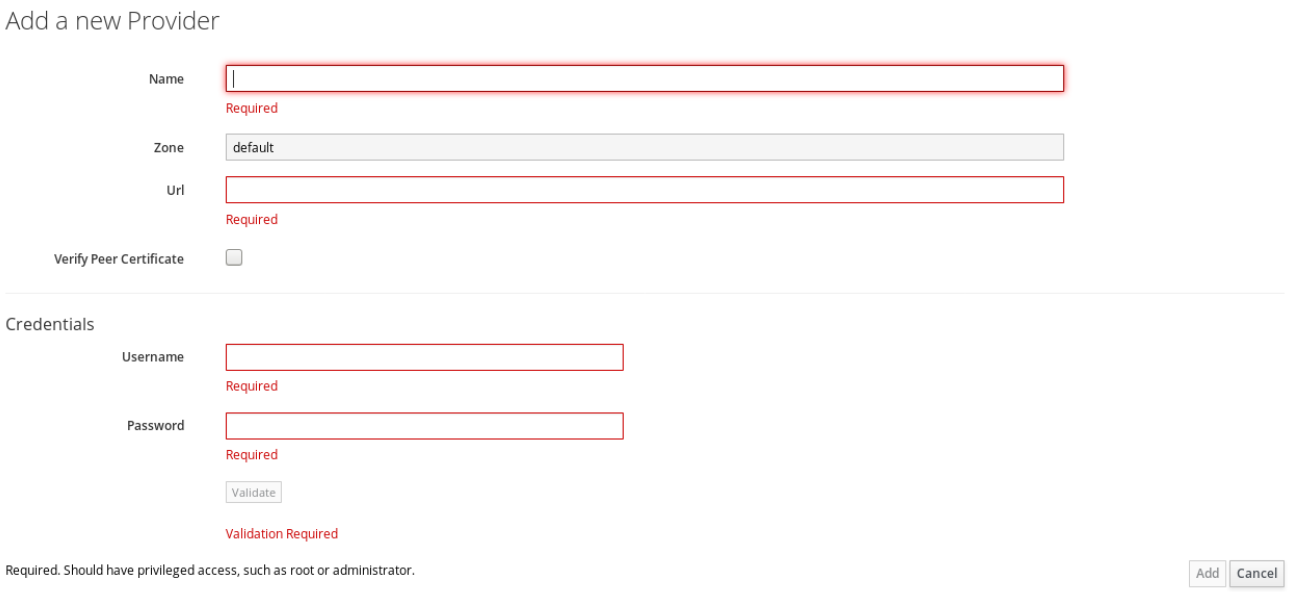
Adding an Ansible Tower Provider

To access your Ansible Tower inventory from Infrastructure Automation, you must add Ansible Tower as a provider.

Notes:

- Ensure **ENABLE HTTP BASIC AUTH** is set to **On** in the Ansible Tower configuration settings before adding the provider. See Tower Configuration in the Ansible Tower *Administration Guide*.
- A trailing slash is **not** required at the end of the Ansible Tower provider URL. Adding the trailing slash to the provider URL may result in a validation error.

1. Browse to menu: **Automation > Ansible Tower > Explorer** and click on the **Providers** accordion tab.
2. Under **Configuration**, click **Add a new Provider**.
3. In the **Add a new Provider** area:



1. Enter a **Name** for the new provider.
2. Add a **Zone** for the provider.
3. Enter the **URL** location or IP address to the Ansible Tower server. Add a trailing slash to the end of the Ansible Tower provider URL.

4. Select the **Verify Peer Certificate** checkbox if desired.
5. In the **Credentials** area, provide the **Username** and **Password**, and **Confirm Password**.
6. Click **Validate** to verify credentials.
7. Click **Add**.

After adding the Ansible Tower provider, refresh its relationships and power states in order to view the current inventory.

Refreshing an Ansible Tower Provider

Refresh relationships of all items related to an existing Ansible Tower configuration management provider including inventory, hosts, virtual machines, and clusters.

You can refresh inventory from Infrastructure Automation, or by enabling the **Update on Launch** option for inventory groups in Ansible Tower. The **Update on Launch** option allows Ansible Tower to automatically update inventory using a dynamic inventory script before launching an Ansible Tower job from a playbook. For more information, see the Ansible Tower documentation.

Important: It can take a long time to retrieve information from providers containing many virtual machines or instances. The Ansible Tower dynamic inventory script can be modified to limit updates to specific items and reduce refresh time.

To refresh an Ansible Tower provider's inventory in Infrastructure Automation:

1. Browse to menu: **Automation > Ansible Tower > Explorer** and click the **Providers** accordion tab.
2. Select the checkboxes for the Ansible Tower providers to refresh under **All Ansible Tower Providers**.
3. Click **Configuration**, and then (**Refresh Relationships and Power States**).
4. Click **OK**.

Infrastructure Automation then queries the Ansible Tower API and obtains an inventory of all available hosts, job, and workflow templates.

Viewing Ansible Tower Providers and Inventory

Infrastructure Automation automatically updates its inventory from Ansible Tower. This includes system groups (known as Inventories in Ansible Tower), basic information about individual systems, and available Ansible Tower job or workflow templates to be executed from the service catalog or Automate.

Note: To view and access Ansible Tower inventories and job or workflow templates in Infrastructure Automation, you must first create them in Ansible Tower.

To view a list of Ansible Tower providers and inventory:

1. Browse to menu: **Automation > Ansible Tower > Explorer**.
2. select the **Providers** accordion menu to display a list of **All Ansible Tower Providers**.
3. Select your Ansible Tower provider to expand and list the inventory groups on that Ansible Tower system. The inventory groups can be expanded to view the systems contained within each group, as well as configuration details for these systems.

Similarly, all discovered job and workflow templates are accessed under the provider by expanding the menu: **Automation > Ansible Tower > Explorer** and click the **Templates** accordion menu.

Viewing Ansible Tower Configured Systems

To view the systems in your Ansible Tower inventory:

1. Browse to menu: **Automation > Ansible Tower > Explorer** and click **Configured Systems**.
2. Under **All Ansible Tower Configured Systems**, select **Ansible Tower Configured Systems** to display a list.

Executing an Ansible Tower Job or Workflow Template from a Service Catalog

You can execute an Ansible Tower playbook from Infrastructure Automation by creating a service catalog item from an Ansible Tower job or workflow template.

Important: You must first create the job or workflow template in Ansible Tower. The job or workflow templates are automatically discovered by Infrastructure Automation when refreshing your Ansible Tower provider's inventory.

First, create a catalog:

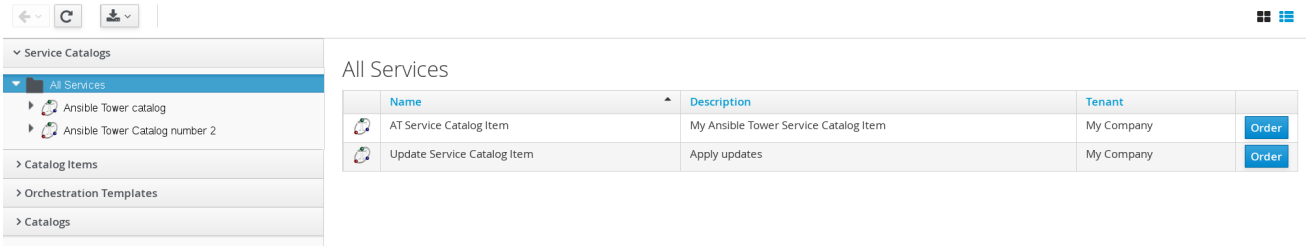
1. Browse to menu: **Services > Catalogs** and click **Catalog Items**.
2. Click **Configuration**, then ⊕ (**Add a New Catalog**)
3. Enter a **Name** and **Description** for the catalog.
4. Click **Add**.

Then, create an Ansible Tower service catalog item:

1. Browse to menu: **Automation > Ansible Tower > Explorer**, then click the **Templates** according menu.
2. Click **Ansible Tower Templates** and select an Ansible Tower job or workflow template.
3. Click **Configuration**, then ⊕ (**Create Service Dialog from this Template**).
4. Enter a **Service Dialog Name** (for example, *ansible_tower_job*)and click **Save**.
5. Browse to menu: **Services > Catalogs** and click **Catalog Items**.
6. Click **Configuration**, then ⊕ (**Add a New Catalog Item**) to create a new catalog item with the following details, at minimum:
    - For **Catalog Item type**, select **Ansible Tower**.
    - Enter a **Name** for the service catalog item.
    - Select **Display in Catalog**.
    - In **Catalog**, select the catalog you created previously.
    - In **Dialog**, select the service dialog you created previously (in this example, *ansible_tower_job*). To ask the user to enter extra information when running the task, **Service Dialog** must be selected. A dialog is required if **Display in Catalog** is chosen.
    - In **Provider**, select your Ansible Tower provider. This brings up the **Ansible Tower Template** option and configures the **Provisioning Entry Point State Machine** automatically.
    - Add configuration information for **Reconfigure Entry Point** and **Retirement Entry Point** as applicable.
    - Select your desired **Ansible Tower Template** from the list. Generally, this is the Ansible Tower job or workflow template previously used to create the service dialog.
7. Click **Add**. The catalog item you created will appear in the **All Service Catalog Items** list.

To execute the Ansible Tower job:

1. Browse to menu: **Service > Catalogs** and click on **Service Catalogs** then click **Ansible Tower catalog**.



2. Click **Order** for the catalog item.
3. Enter any variables requested and click **Submit**.

Infrastructure Automation takes you to the **Requests** queue page and show the status of the job.

The service item's details can be viewed in menu: **Services > My Services** in Infrastructure Automation.

Note: Instead of running a single job at a time, multiple service catalog items can also be grouped together as a catalog bundle to create one deployment with multiple job templates. For more information, see Catalogs and Services.

Executing an Ansible Tower Job Using a Custom Automate Button

Infrastructure Automation can execute Ansible Tower jobs on virtual machines or instances using custom buttons in Automate.

Ansible Tower jobs can either be noncustomizable, which do not require any extra configuration from the user, or alternatively, they can allow the user to specify a parameter (for example, a package name to install). In Ansible Tower jobs containing a dialog, Infrastructure Automation accepts additional information from the user and adds it to the appropriate API call in Automate, and then sends it into Ansible Tower.

Prerequisites.

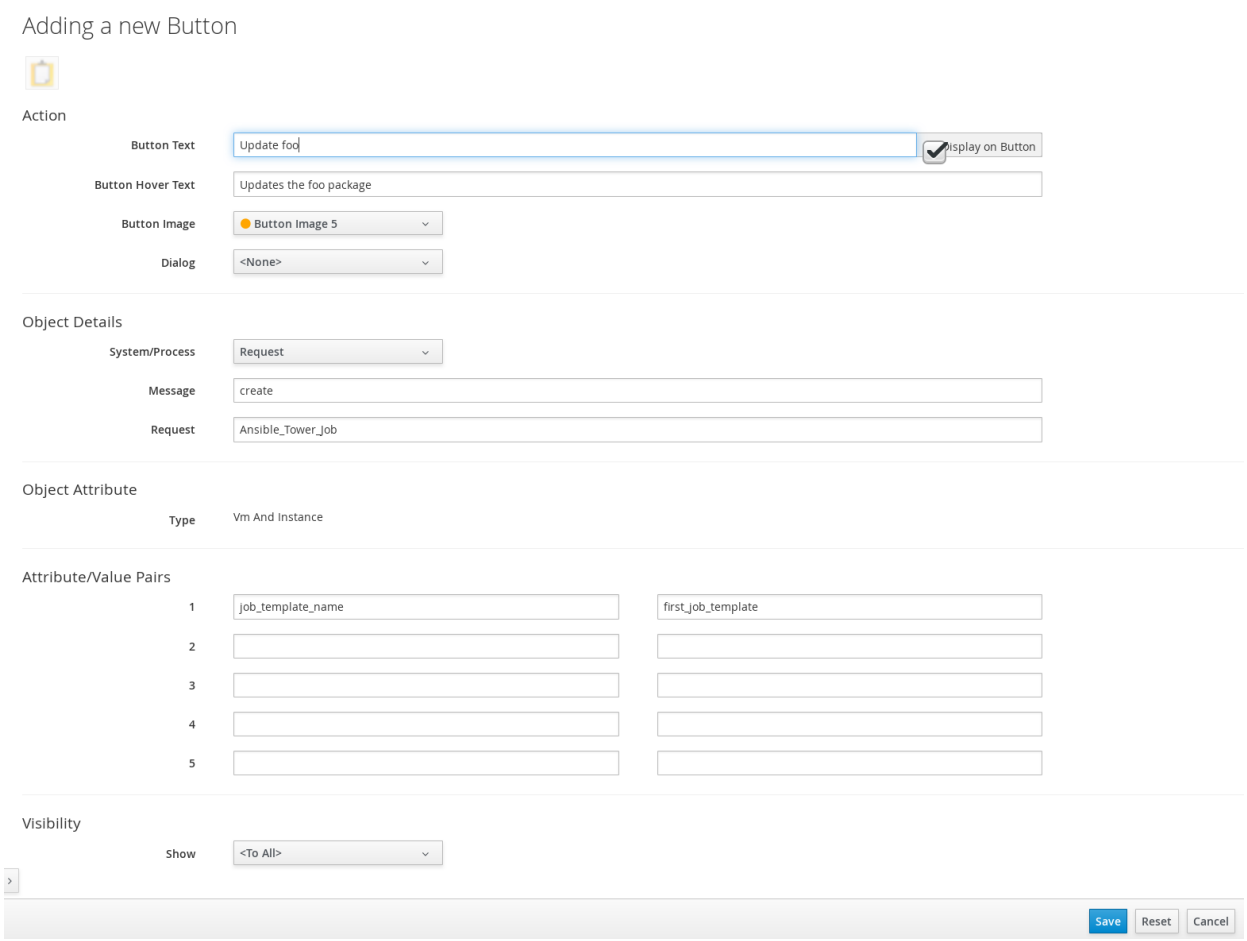
Before creating an Automate button to execute an Ansible Tower job, the following must be configured:

- An Ansible playbook in Ansible Tower. See the Ansible Tower documentation for instructions.
- Ansible Tower must be able to reach virtual machines or instances deployed by Infrastructure Automation at the IP level.
- The virtual machine template must have the Ansible Tower environment's public SSH key injected. For cloud instances, `cloud-init` can be used and the public SSH key can be passed without rebuilding the image.
- Any dynamic inventory scripts used must be configured to return the virtual machine names exactly as they are stored in Infrastructure Automation, without the UUID appended.

Executing an Ansible Tower Job using a Custom Automate Button.

To configure a custom button to execute an Ansible Tower job on a virtual machine or instance, first create the button:

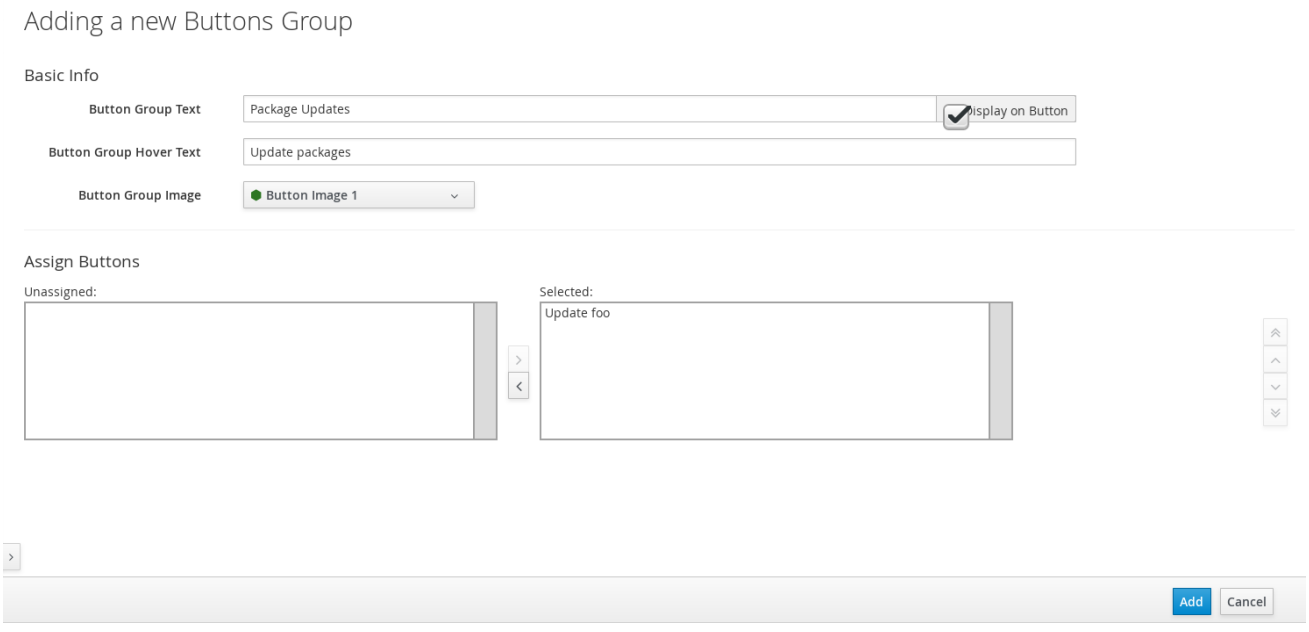
1. Browse to menu: **Automation > Automate > Customization**.
2. Click the **Buttons** accordion menu.
3. Click menu: **VM and Instance > Unassigned Buttons**. This configures the button to run on virtual machines or instances.
4. Click **Configuration**, then click ⊕ (**Add a new Button**).
    1. In the **Adding a new Button** screen, configure the **Action** parameters as desired. **Dialog** can be left blank if the playbook does not require extra variables. To ask the user to enter extra information when running the task, **Service Dialog** must be selected.
    2. Configure **Object Details** fields with the following request details:
    - For **System/Process**, select **Request**.
    - For **Message**, enter **create**.
    - For **Request**, enter **Ansible_Tower_Job**.
    1. Configure **Attribute/Value Pairs** with the following parameters:
    - **job_template_name** is the Ansible Tower job template name to associate with the button. The **job_template_name** field is mandatory; other parameters are provided by the Tower job dialog.
    - Configure **Visibility** to all users, or limit visibility by role as desired.



1. Click **Add**.

If you do not have an existing button group to assign the new button to, create a new button group:

1. From menu: **Automation > Automate > Customization**, browse to menu: **Buttons** and click **VM and Instance > Add a new Button Group**, and configure the following:
    1. Configure **Basic Info** as desired. For example, name the button group `VM Actions`.
    2. In **Assign Buttons**, select the button you just created from the **Unassigned** list and click to assign it to **Selected**.



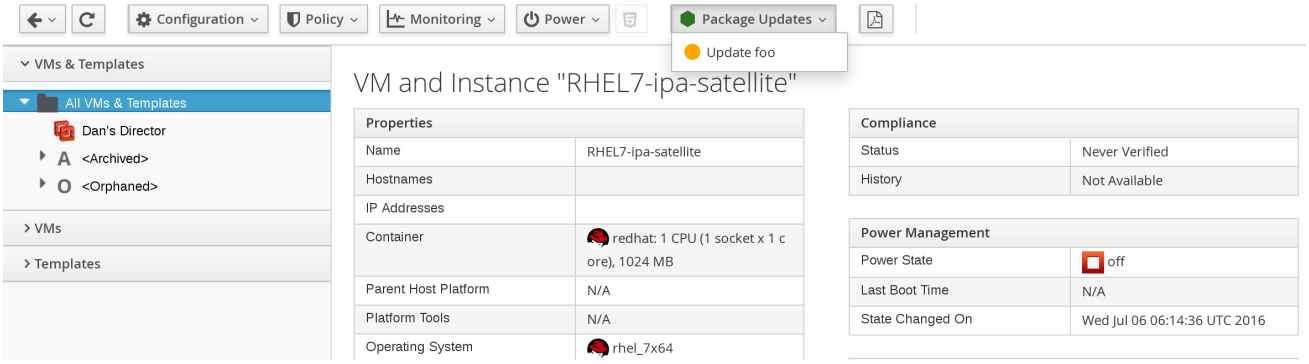
1. Click **Add**.

To assign the button to an existing button group:

1. Browse to menu: **Buttons > VM and Instance > VM Actions > Edit this Button Group**.
2. In **Assign Buttons**, select the button you just created from the **Unassigned** list and click › to assign it to **Selected**.
3. Click **Add**.

To use the button to run an Ansible Tower job on a virtual machine:

1. Browse to menu: **Compute > Infrastructure > Virtual Machines**.
2. Select the virtual machine to run the Ansible Tower job template on.
3. Click the **VM Actions** button to show the button you created, and click the button from the list to run the Ansible Tower job template.



4. Click **Submit** to execute the job.

Infrastructure Automation then confirms the job has been executed.

If you selected a service dialog to run when creating the button, Infrastructure Automation will then prompt you to enter variables to complete the task. After entering your desired parameters, Infrastructure Automation takes you to the **Requests** page.

The service item's details can be viewed in menu: **Services > My Services** in Infrastructure Automation.

Cloud Providers

In Infrastructure Automation, a cloud provider is a cloud computing environment that you can add to a Infrastructure Automation appliance to manage and interact with the resources in that environment. This chapter describes the different types of cloud providers that you can add to Infrastructure Automation, and how to manage them. Most cloud providers are added individually to Infrastructure Automation. Additionally, Amazon EC2 and Azure cloud providers can be discovered automatically by Infrastructure Automation.

The web interface uses virtual thumbnails to represent cloud providers. Each thumbnail contains four quadrants by default, which display basic information about each provider:

1. Number of instances
2. Management system software
3. Number of images
4. Authentication status

Icon	Description
	Validated: Valid authentication credentials have been added.

Icon	Description
!	Invalid: Authentication credentials are invalid.
?	Unknown: Authentication status is unknown or no credentials have been entered.

Provider authentication status

Refreshing Cloud Providers

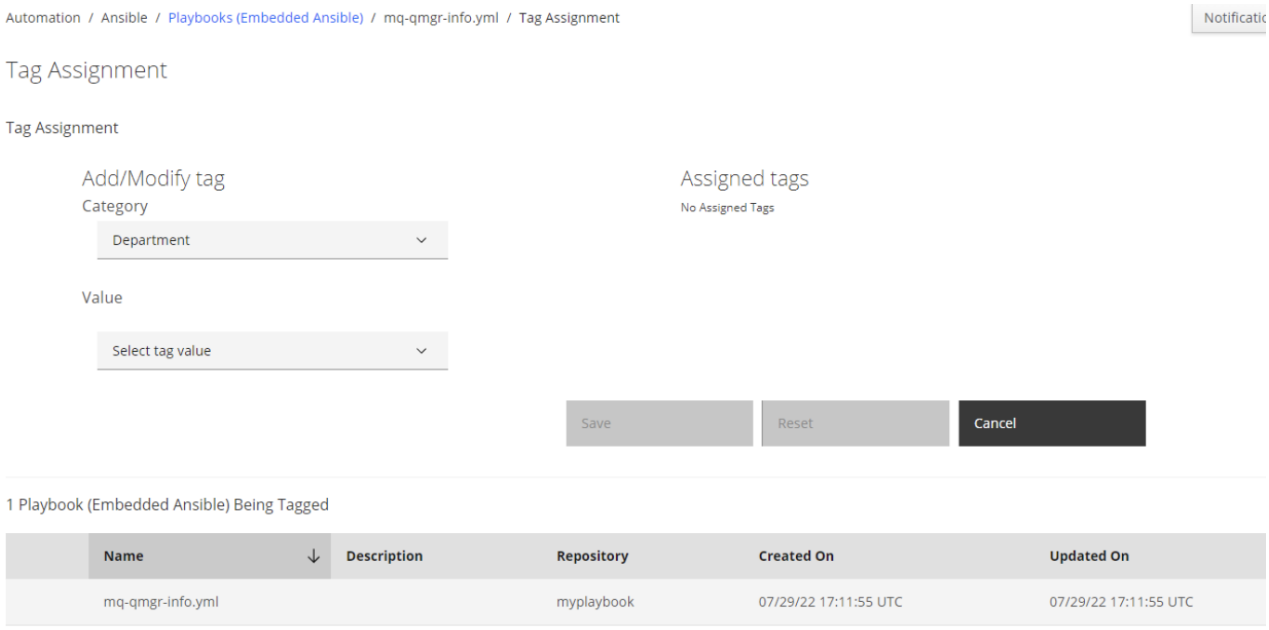
Refresh a cloud provider to find other resources related to it. Ensure the chosen cloud providers have the correct credentials before refreshing.

1. Browse to menu: **Compute > Clouds > Providers**.
2. Select the checkboxes for the cloud providers to refresh.
3. Click **Configuration**, and then (**Refresh Relationships and Power States**).
4. Click **OK**.

Tagging Cloud Providers

Apply tags to all cloud providers to categorize them together at the same time.

1. Navigate to **Compute > Clouds > Providers**.
2. Select the checkboxes for the Cloud Providers to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. Select a customer tag to assign from the first list.



5. Select a value to assign from the second list.
6. Click **Save**.

Removing Cloud Providers

A cloud provider might require removal from the VMDB if it is no longer in use.

1. Browse to menu: **Compute > Clouds > Providers**.
2. Check the cloud providers to remove.
3. Click **Configuration**, and then (**Remove Cloud Providers from the VMDB**).
4. Click **OK**.

Editing a Cloud Provider

Edit information about a provider such as the name, IP address, and login credentials.

Note:

The **Type** value is unchangeable.

To use a different cloud provider, create a new one.

1. Browse to menu: **Compute > Clouds > Providers**.
2. Click the cloud provider to edit.
3. Click **Configuration**, and then **(Edit Selected Cloud Provider)**.
4. Edit the **Basic Information**. This varies depending on the **Type** of provider.
5. Fill out the **Credentials** by typing in a **Username**, **Password**, and a verification of this password (**Confirm Password**).
    - If selecting **Amazon EC2**, generate an **Access Key** in the **Security Credentials** of your Amazon AWS account. The **Access Key ID** acts as your **User ID**, and your **Secret Access Key** acts as your **Password**.
    - If selecting **OpenStack**, use the `Keystone User ID` and `Password` for your login credentials.
6. If editing an OpenStack provider, use the **AMQP** subtab to provide credentials required for the Advanced Message Queuing Protocol service on your OpenStack Nova component.
7. Click **Validate** and wait for notification of successful validation.
8. Click **Save**.

Viewing a Cloud Provider's Timeline

View the timeline of events for instances registered to a cloud provider.

1. Navigate to **Compute > Clouds > Providers**.
2. Click the desired cloud provider for viewing the timeline.
3. Click (**Monitoring**), and then (**Timelines**).
4. From **Options**, customize the period of time to display and the types of events to see.
    - Use **Show** to select regular Management Events or Policy Events.
    - Use the **Type** list to select hourly or daily data points.
    - Use **Date** to type the date for the timeline to display.
    - If you select to view a daily timeline, use **Show** to set how many days back to go. The maximum history is 31 days.
    - The three **Event Groups** list allow you to select different groups of events to display. Each has its own color.
    - From the **Level** list, select a **Summary** event, or a **Detail** list of events.

- Azure Providers
- Amazon EC2 Providers
- Google Compute Engine Providers

- IBM Cloud VPC Provider
- IBM Power Systems Virtual Servers Providers
- IBM PowerVC Providers
- OpenStack Providers
- Oracle Cloud Providers

Physical Infrastructure Providers

In Infrastructure Automation, a physical infrastructure provider is a management console used to manage physical inventory such as physical servers, chassis, racks, and physical networking/storage devices.

ManageIQ supports the following types of physical infrastructure providers:

- Lenovo XClarity
- Redfish

Image Import Workflows

IBM PowerVC to IBM Power Systems Virtual Server

The image import functionality allows you to easily move an image from your on-prem PowerVC environment to an off-prem Power Systems Virtual Server environment using Infrastructure Automation web interface. The enablement of the workflow in Infrastructure Automation requires you to perform certain preparations that are described in detail below.

Preparing Infrastructure Automation hosting server (required once):

The following commands are to be run on a server (or in a corresponding docker container) that is dedicated to hosting your Infrastructure Automation instance.

1. Install **`ansible-runner`** application using the installation instructions. The application must be available in the the command line of the user under which the Infrastructure Automation web-server is running.

Preparing IBM PowerVC Server (required once):

The following commands are to be run on a PowerVC server that is dedicated to supporting the image import workflow.

1. Make sure that the minimal IBM PowerVC version you are using is 1.4.4.
2. Install **`python3.X`**, **`pip3`**, **`virtualenv`** applications as well as the **`libselinux-python3`** library. You can achieve this by issuing the following command as a “root” user:

   ```
   yum install -y python3 python-pip3 python-virtualenv libselinux-python3 gcc
   ```

   The standard **`python3.X`** installation will be used in order to run auxiliary scripts, which will upload PowerVC disk images (in Open Virtualization Format) to the IBM Cloud Object Storage. The **`pip3`** is required in order to install Cloud Object Storage Python SDK and auxiliary packages for proper functioning of the import workflow. The role of **`virtualenv`** will be to isolate python package installations that are relevant only to the image import workflow from the Python system packages. The **`libselinux-python3`** library is needed for the proper functioning of Ansible’s **`copy`** module, which will be used for the copying of auxiliary data between the Infrastructure Automation hosting site and the PowerVC server. Presence of **`gcc`** is needed as a dependency of one of the Python packages (**`pycryptodome`**) that will be installed on later steps.
3. Prepare enough local storage for holding transient PowerVC disk images that will be transported into Power Systems Virtual Server and deleted from a local storage upon workflow completion. A typical PowerVC image file size can be many GBs. As an example, a RHEL 8.3 disk image can be of 20GB size or more. You would therefore require more space in your local storage than the mentioned disk image size in order to support the import workflow. You would also need enough space to hold Python virtual environment described in the next steps. Its size however will be no more than 100MB. Additionally, import session will place auxiliary files into the sessions directory during the workflow execution but their collective size will not exceed 5MB. In summary, if you’re attempting to import an image whose size is ‘G’ Gigabytes, then having ‘G+1’ Gigabytes of free space in that directory would be sufficient.
4. Create an empty sessions directory **`/home/sessions`** which will have to be located on the local storage with sufficient space as described in the previous step. In addition, make sure that already existing **`/var/opt/ibm/powervc/imgstaging`** directory also has at least G+1

Gigabytes of free storage as described in previous paragraph since IBM PowerVC internally uses it as a temporary staging area during export of images to the local filesystem.

```
mkdir /home/sessions
```

5. The image import workflow will use Ansible via SSH from the Infrastructure Automation server in order to execute commands on the PowerVC server. At this point you need to choose a system user that will be made available to Infrastructure Automation instance for connection to the PowerVC server via SSH. Make sure the user has a read/write access to the **/home/sessions** directory. The user must also be able to execute **powervc-image** application through the command line. In general, a good security practice would be to permit user only as much as it is required to run the import workflow. We will assume that all further commands in this section are executed using this existing or newly created system user:

   ```
   su - your_user
   ```

6. Create a separate virtual Python environment:

   ```
   virtualenv --system-site-packages /home/sessions/venv
   ```

   This environment will hold Python packages that are relevant only to the image import workflow and as such will not be available system-wide. Conversely, the **--system-site-packages** will indicate to the application that the existing Python system packages are to be made available to the virtual environment.

7. Installing packages in the environment requires one to first activate the environment after which all the Python package installations via **pip3** will become environment relative only. Therefore, activate the newly created Python environment:

   ```
   source /home/sessions/venv/bin/activate
   ```

8. Install Python packages inside the virtual environment using:

   ```
   pip3 install pathlib pycryptodome ibm-cos-sdk
   ```

   The **pathlib** is used for handling file paths. The **pycryptodome** is required for encrypting/decrypting IBM Cloud connection credentials and the **ibm-cos-sdk** represents the IBM Cloud Object Storage Python SDK.

9. Upon completing the Python package installations above you should exit the virtual environment by running:

   ```
   deactivate
   ```

10. Make sure to have a valid PowerVC connection/resource file under **/opt/ibm/powervc/powervcrc** (or at your custom location) containing similar definitions (the values below represent an example and not a suggestion for usage).

    ```
    export OS_IDENTITY_API_VERSION=3
    export OS_AUTH_URL=https://host:5000/v3/
    export OS_CACERT=/etc/pki/tls/certs/powervc.crt
    export OS_REGION_NAME=RegionOne
    export OS_PROJECT_DOMAIN_NAME=Default
    export OS_PROJECT_NAME=ibm-default
    export OS_TENANT_NAME=$OS_PROJECT_NAME
    export OS_USER_DOMAIN_NAME=Default
    export OS_USERNAME=root
    export OS_PASSWORD=password
    export OS_COMPUTE_API_VERSION=2.46
    export OS_NETWORK_API_VERSION=2.0
    export OS_IMAGE_API_VERSION=2
    export OS_VOLUME_API_VERSION=2
    ```

11. The last step is to test the exporting of PowerVC disk image to the local filesystem. Firstly, execute:

    ```
    source /opt/ibm/powervc/powervcrc
    ```

    This will acquire all the environmental variables required for issuing commands to the corresponding PowerVC instance. Now run the following command:

    ```
    powervc-image list
    ```

    The output should be similar to the one below:

    ```
    +----------+--------------------------------------+--------+---------+------+-------------+-------------
    -+-----------+------------+
    | Name     | ID                                   | Status | Volumes | Size | Description | Architecture
    | OS Distro | Exportable |
    +----------+--------------------------------------+--------+---------+------+-------------+-------------
    -+-----------+------------+
    | test-IMG | XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX | active | 1       | 20   |             | ppc64
    | rhel      | True       |
    +----------+--------------------------------------+--------+---------+------+-------------+-------------
    -+-----------+------------+
    ```

    Note the name of the image you would like to export, which will be **test-IMG** in our example. Next export the image of your choice:

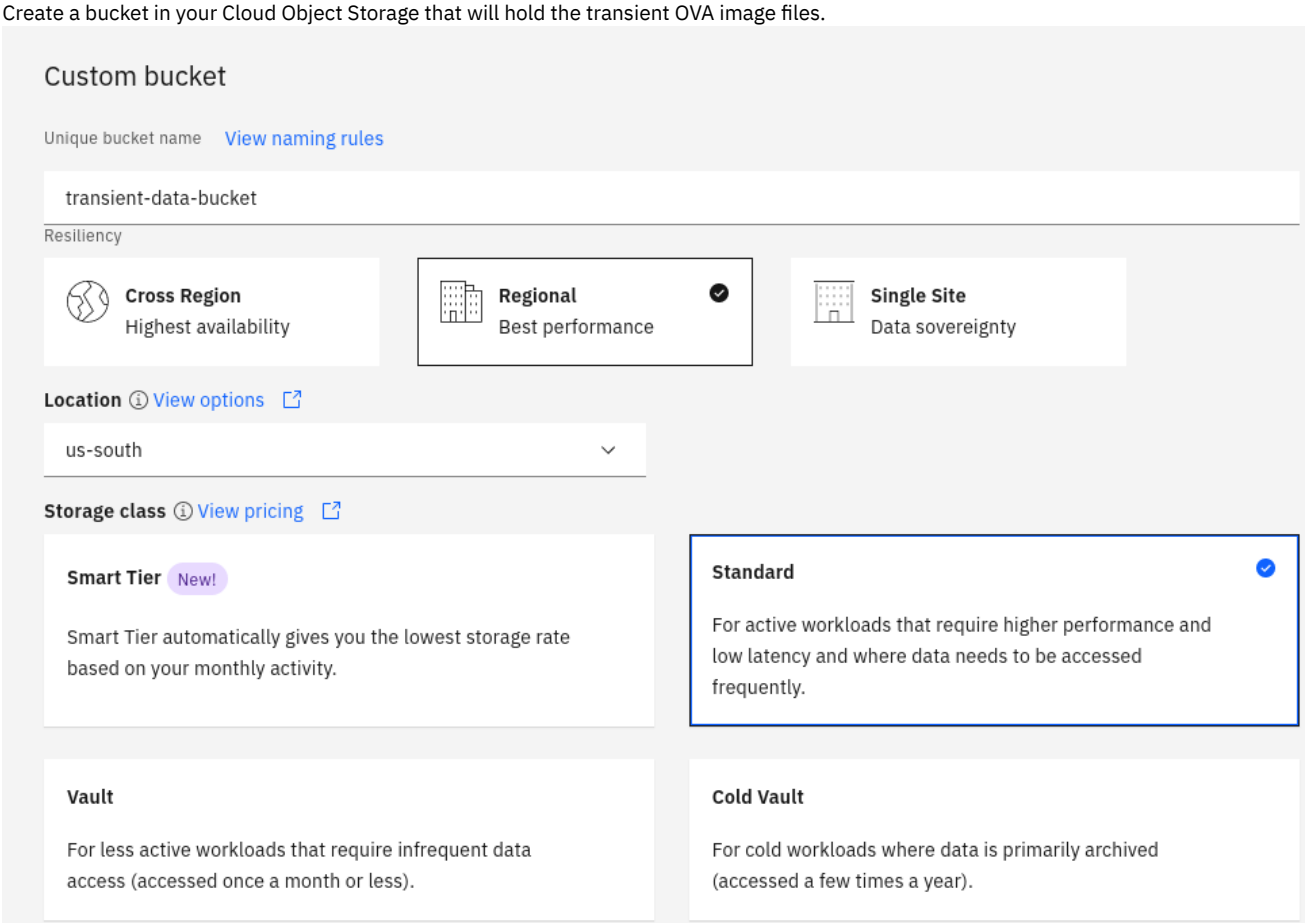
```
powervc-image export -i test-IMG -p /home/sessions/image.ova
```

It is a time-consuming operation with the runtime taking several minutes to complete depending on the size of exported image. Once the operation completes you should be able to find the exported image under **`/home/sessions/image.ova`**. At this stage you can remove the exported image thus concluding the test:

```
rm /home/sessions/image.ova
```

Add a Cloud Object Storage provider in Infrastructure Automation:

1. See the corresponding instructions here.
2. Create a bucket in your Cloud Object Storage that will hold the transient OVA image files.



Make sure to choose the region for the bucket that **corresponds** to the region that you have specified when registering the Cloud Object Storage provider.

NOTE: The “root” user as well as the workflow dedicated user of the PowerVC server from step 5 would both be able to decrypt the Cloud Object Storage’s credentials during the execution of the import workflow and therefore make sure to take this into consideration in your security setup.

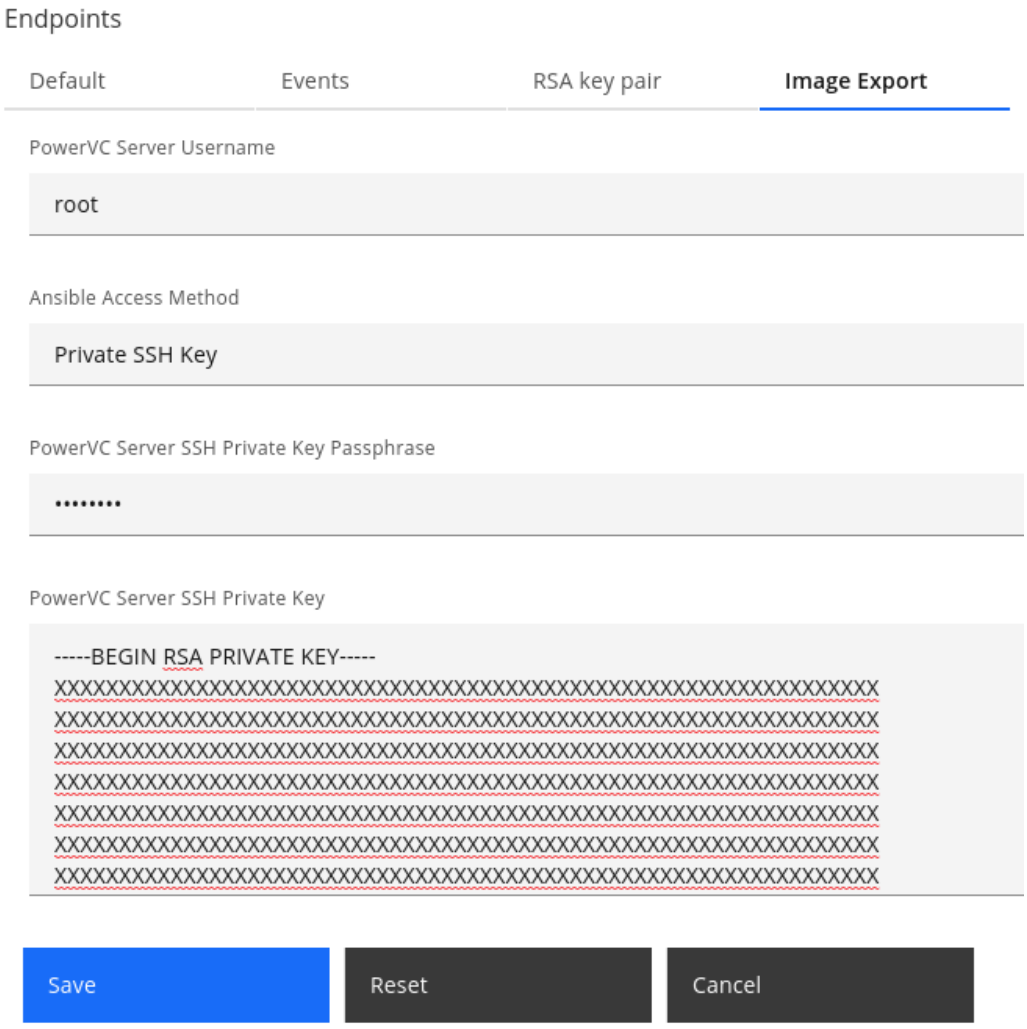
Add an IBM Power Systems Virtual Server provider in Infrastructure Automation:

1. See the corresponding instructions here.

Add an IBM PowerVC provider in Infrastructure Automation:

1. See the corresponding instructions here.
2. Specify the image import related parameters on the PowerVC registration form.
   - Enter the SSH username for access to the PowerVC server. It corresponds to the user on your PowerVC server that is dedicated to running image import related commands via Ansible as described in 5.

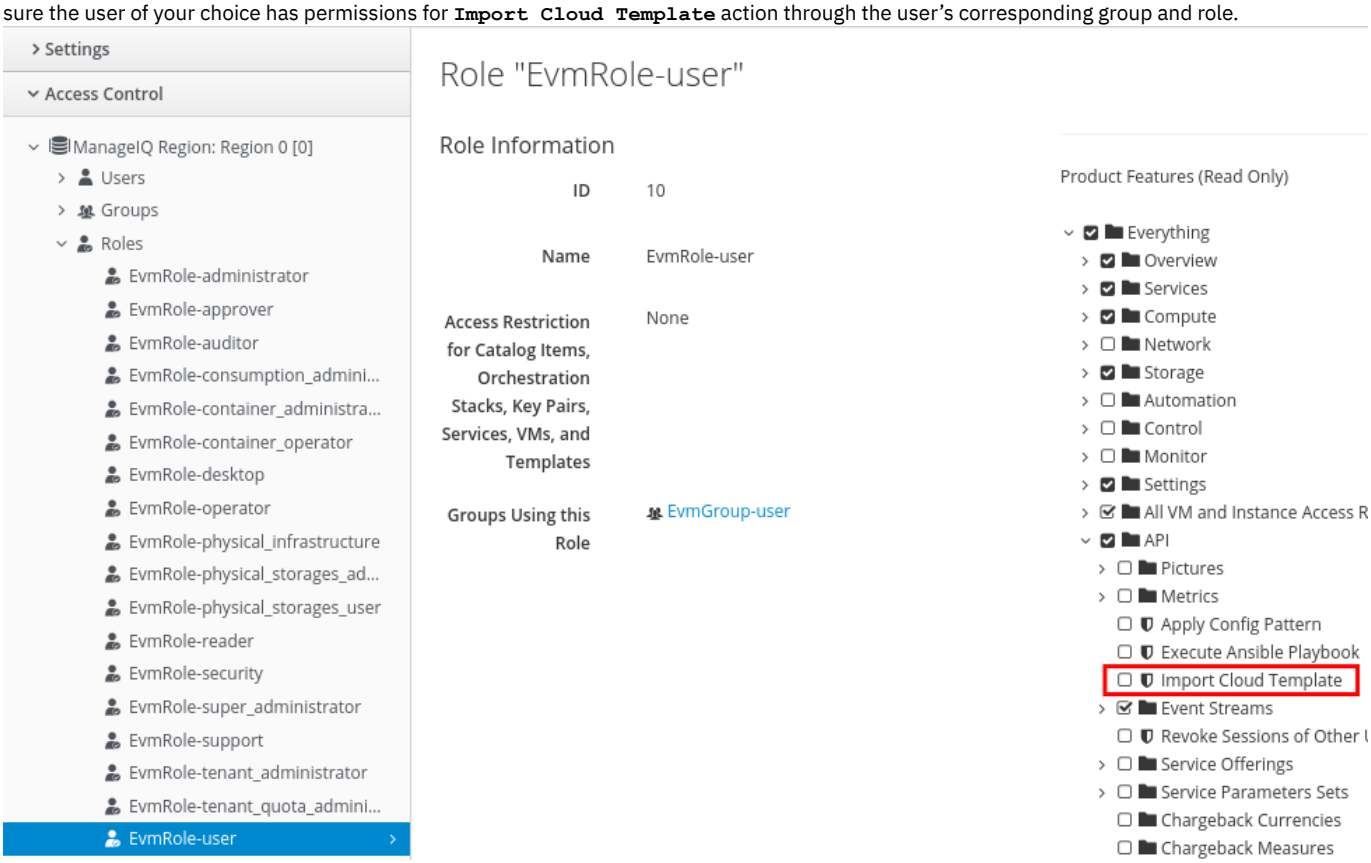
- Enter the private key and the private key passphrase (if present).



Grant Image Import Permissions in Infrastructure Automation:

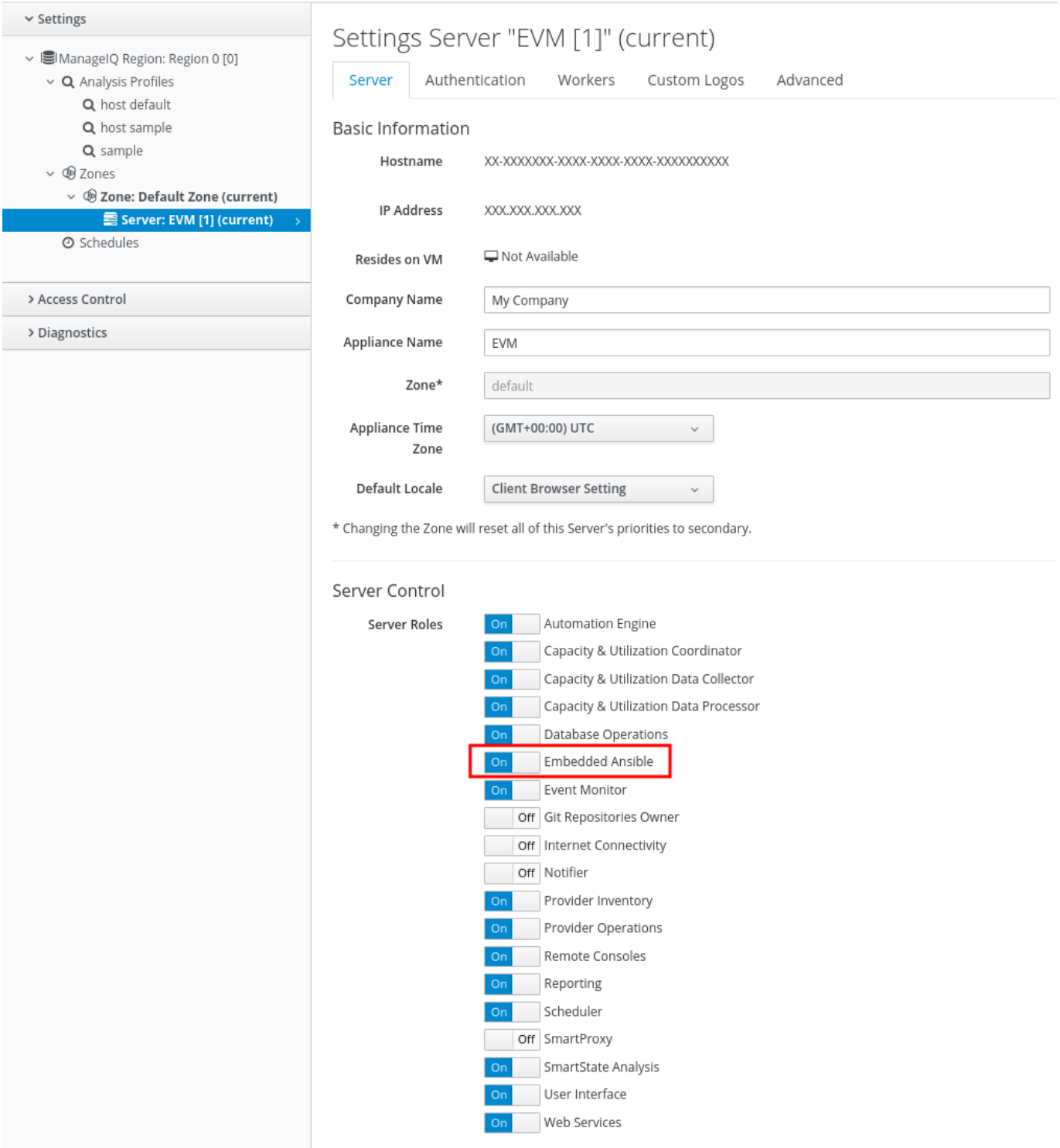
User performing image import needs a corresponding permission in Infrastructure Automation in order to perform this operation. For granting permissions, log-in through an administrative account and navigate to **`Settings -> Application Settings -> Access Control`**. Make

sure the user of your choice has permissions for **`Import Cloud Template`** action through the user's corresponding group and role.



Enable Embedded Ansible in Infrastructure Automation:

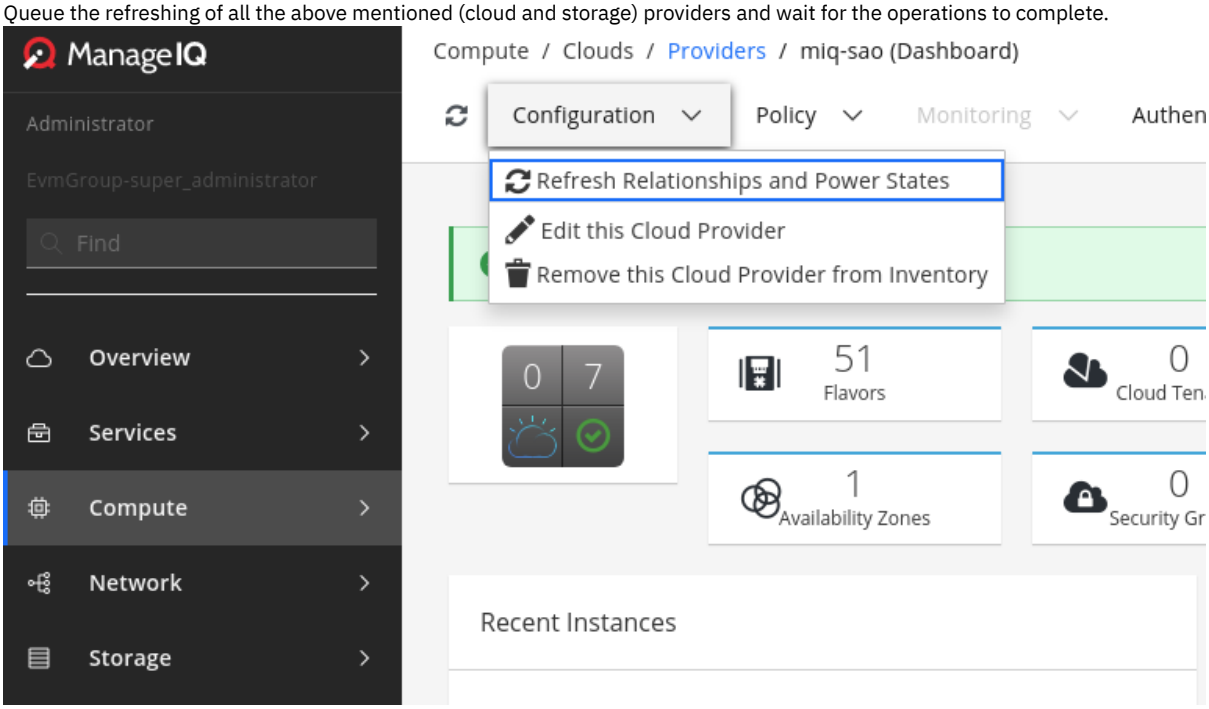
Currently the workflow event is queued as Embedded Ansible playbook upon request submission and as such the corresponding functionality has to be enabled in Infrastructure Automation by navigating to: **`Settings -> Application Settings -> Settings`**



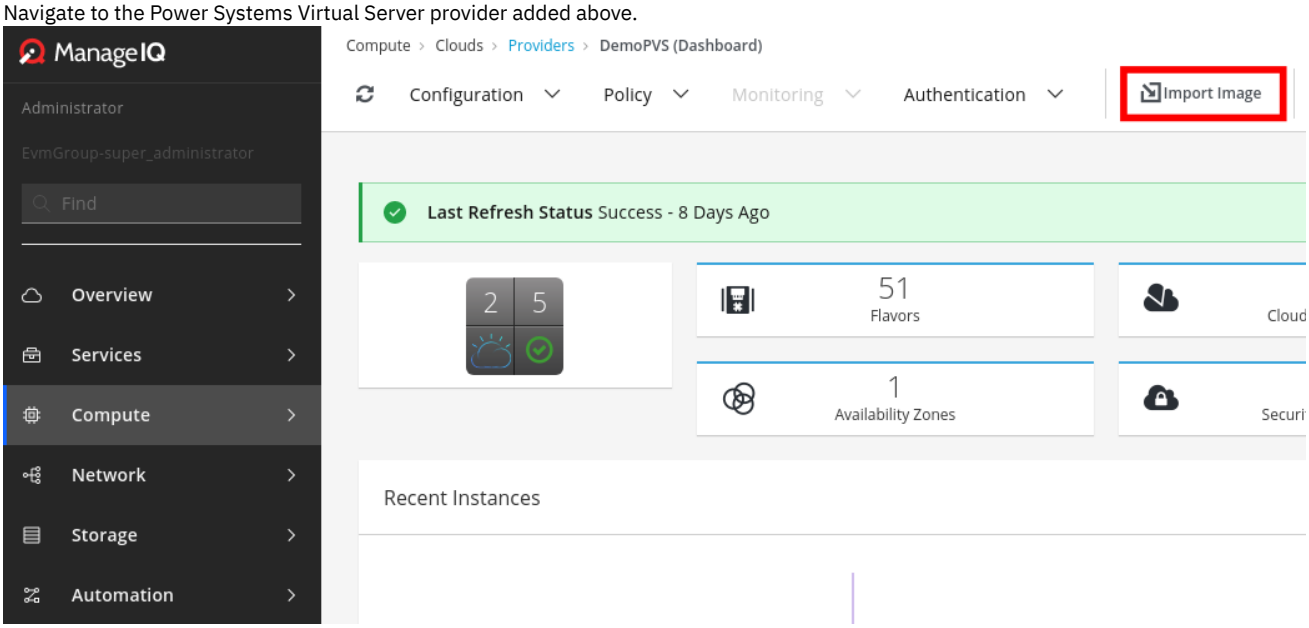
Start the workflow

The workflow operates by exporting the image as an OVA file onto a PowerVC local storage, uploading it to the Cloud Object Storage bucket and then transferring it into Power Systems Virtual Server image registry. In order to initiate the workflow:

1. Queue the refreshing of all the above mentioned (cloud and storage) providers and wait for the operations to complete.

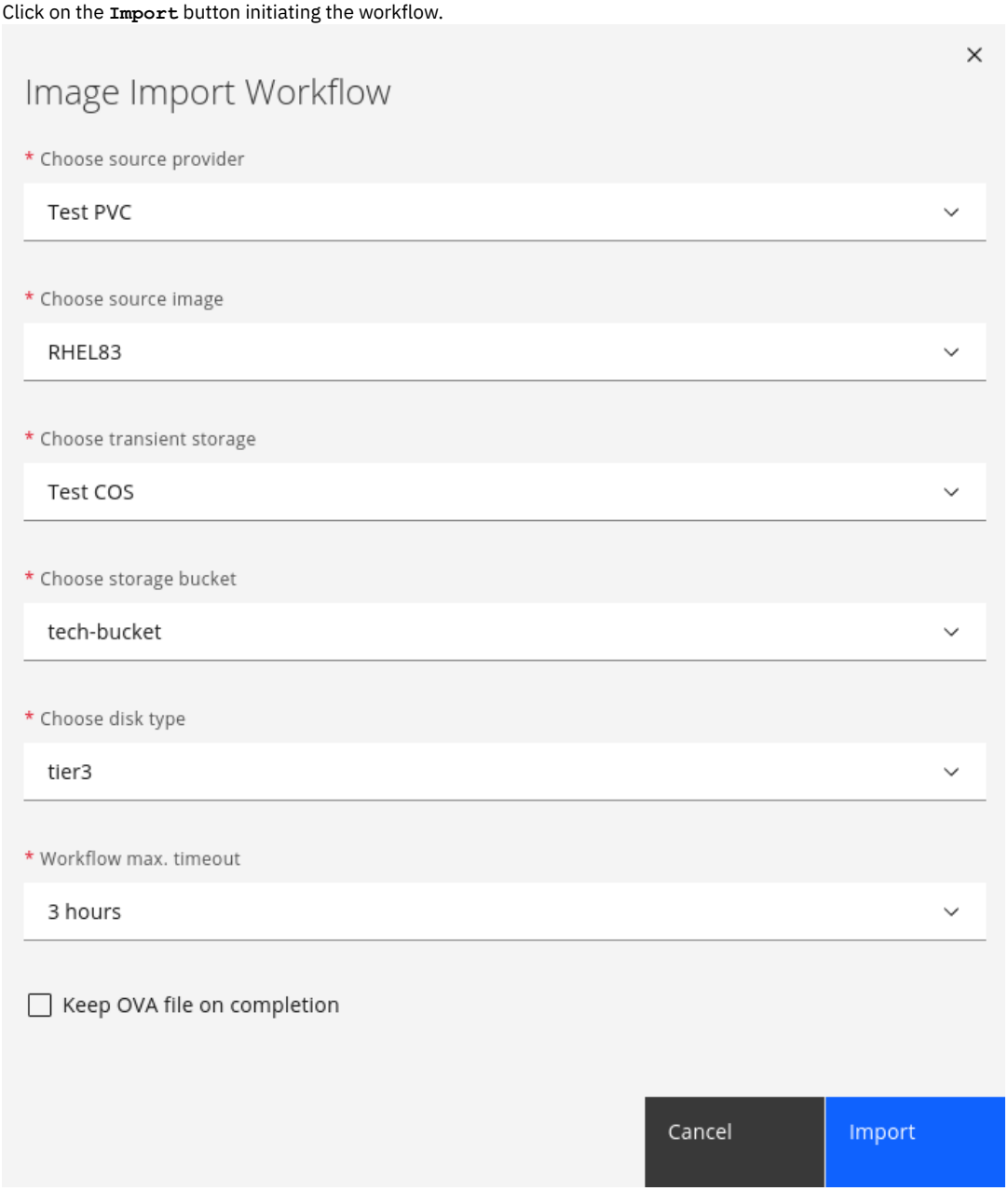


2. Navigate to the Power Systems Virtual Server provider added above.



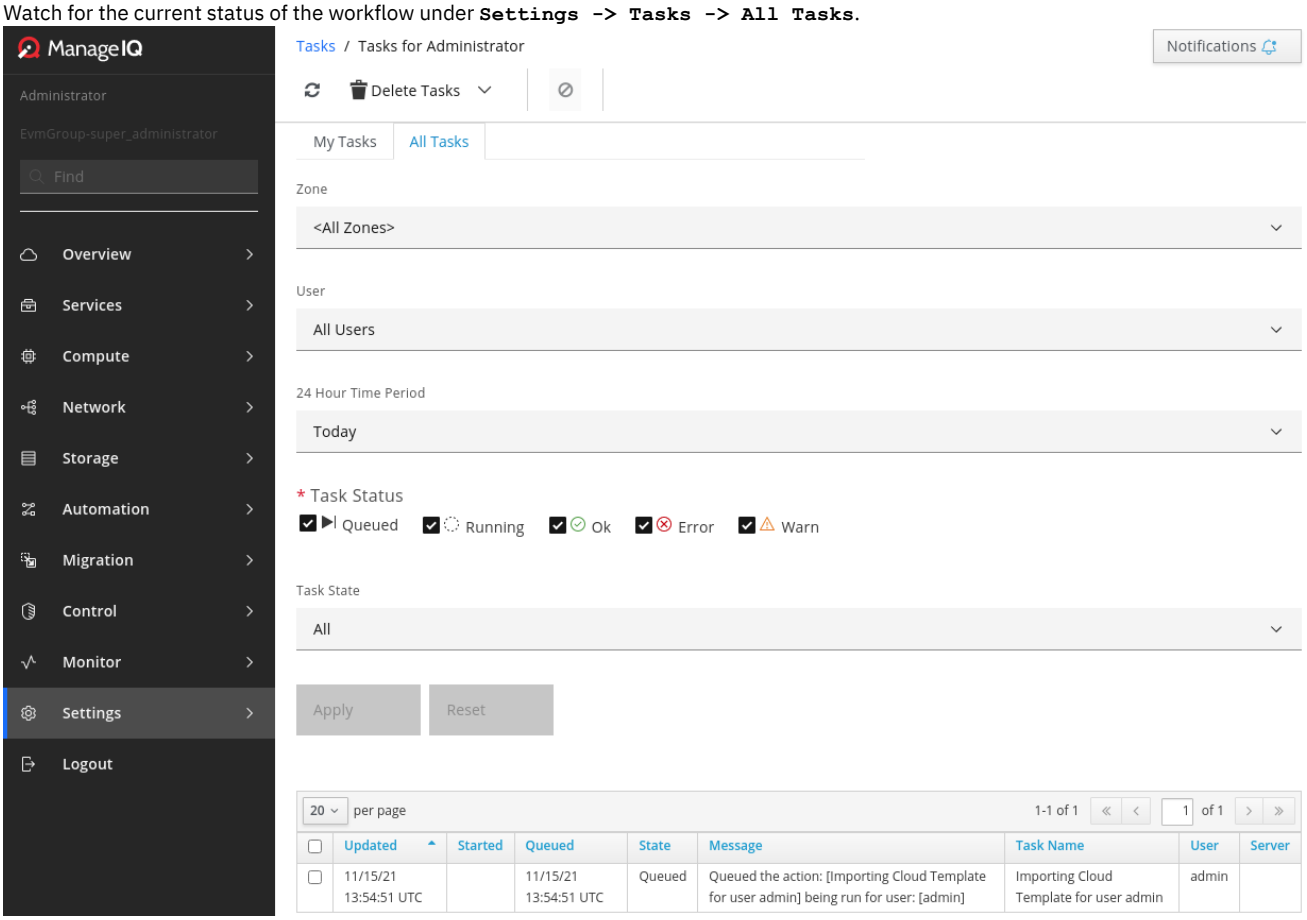
3. Click on the **`Import Image`** button after which an import form will appear.
4. Choose the PowerVC provider added above as a source provider.
5. Choose the image you would like to import.
6. Choose the Cloud Object Storage provider added above.
7. Choose the transient bucket created above.
8. Choose target disk type for the image being imported.
9. Choose the timeout value for your request (default is 3 hours).
10. Check the corresponding box if you wish to keep the OVA file in the cloud bucket upon workflow completion.

11. Click on the **`Import`** button initiating the workflow.



12. The workflow can take an hour or more depending on the image size, bucket location, Power Systems Virtual Server and PowerVC internal load.

13. Watch for the current status of the workflow under **`Settings -> Tasks -> All Tasks`**.



14. Initiate the refreshing of the Power Systems Virtual Server provider upon workflow completion and wait for the newly imported image to appear.

Troubleshooting

- If the Infrastructure Automation UI shows no detailed error description then make sure to check the latest logs with **`journalctl -t evm`** or **`oc logs`** on a server or pod that hosts your Infrastructure Automation instance.
- If **`Infrastructure Automation`** server is terminated during the workflow execution then certain residual objects will have to be removed manually. These are potentially the session directory on the PowerVC server under **`/home/sessions`**, the already uploaded OVA image file in the Cloud Object Storage bucket or its parts as explained here (see **`Abort incomplete multipart uploads`**).
- Currently only one import workflow for a given PowerVC <–> Power Systems Virtual Server pair is allowed. Any newly submitted import workflow, where at least one of the instances in the pair is involved with some other import workflow request, will be queued until the latter completes. You would need to cancel such already executing workflow in order to force immediate execution of the newly submitted one.

Network Managers

In Infrastructure Automation, a network manager is an inventory of networking entities on existing cloud and infrastructure providers managed by your Infrastructure Automation appliance.

This provider type exposes software-defined networking (SDN) providers including *OpenStack Network (Neutron)*, *Azure Network*, and *Amazon EC2 Network*, which enables software-defined networking inventory collection. The OpenStack Network provider collects inventory of floating IPs from OpenStack so that IPs can be allocated without querying OpenStack database every time. Also, it refreshes all Neutron data from both OpenStack and OpenStack Infrastructure, and extracts the Neutron logic to a shared place. Note that management via the network providers configuration is currently disabled.

This chapter describes the different types of network managers available to Infrastructure Automation, and how to manage them. Network managers are discovered automatically by Infrastructure Automation from other connected providers.

Adding or Viewing Network Providers

Note:

All supported network providers—OpenStack Network, Azure Network, and Amazon EC2 Network, are added or removed automatically upon adding or removing the respective cloud provider.

Viewing network providers:

1. Browse to menu: **Networks > Providers** to see a list of all network providers, along with information such as *Name*, *Type*, *EVM Zone*, *Number of Instances*, *Subnets*, and *Region*.
2. Click on a provider from the list to view its summary screen.

Network providers summary:

The summary screen includes tables containing information on *Properties*, *Status*, *Relationships*, *Overview*, and *Smart Management*. Click on rows in the *Relationship* and *Overview* tables to see detailed information for individual entities.

Accordion tabs in the sidebar provide access to **Properties** and **Relationships** details.

Click on **Reload**, **Configuration**, **Policy**, and **Monitoring** actions in the taskbar to manage the selected provider.

Note:

Alternatively, click on a cloud provider to see the cloud provider details and its relationships such as Network Manager, Tenants, Instances among others. In Relationships, click Network Manager to see information about the network provider, and its relationship with the cloud provider, on the summary page.

Refreshing Network Providers

Refresh a network provider to find other resources related to it. Ensure the selected network providers have the correct credentials before refreshing.

1. Browse to menu: **Networks > Providers**.
2. Select the network providers to refresh.
3. Click **Configuration**, and then (**Refresh Relationships and Power States**).
4. Click **OK**.

Tagging Network Providers

Apply tags to network providers to categorize them together at the same time.

1. Navigate to **Networks > Providers**.
2. Select the network providers to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. **Select a customer tag to assign** from the first list.
5. Select a value to assign from the second list.
6. Click **Save**.

Removing Network Providers

Although network providers are added or removed automatically upon adding or removing the respective cloud provider, you can manually remove a network provider if it is no longer in use. This will remove the network provider from the VMDB and any relationship with the cloud provider.

1. Browse to menu: **Networks > Providers**.
2. Click the network provider to remove.

3. Click **Configuration**, and then (**Remove this Network Provider from the VMDB**).
4. Click **OK**.

Viewing a Network Provider's Timeline

View the timeline of events for instances registered to a network provider.

1. Navigate to **Networks > Providers**.
2. Click the network provider you want to monitor the timeline for.
3. Click (**Monitoring**), and then (**Timelines**).
4. From **Options**, select the event type and interval, and customize the period of time to display and the types of events to see.
    - Select *Management Events* or *Policy Events* from the **Show** list.
    - Select an **Interval** between *Hourly* and *Daily*.
    - Select **Date**.
    - If you selected *Daily* for **Interval**, set the number of days in the past to see the event timeline for. The maximum is *31 days back*.
    - Select **Summary** or **Detail** for **Level**.
    - Select the required **Event Groups** from the lists you want to monitor the timeline for.

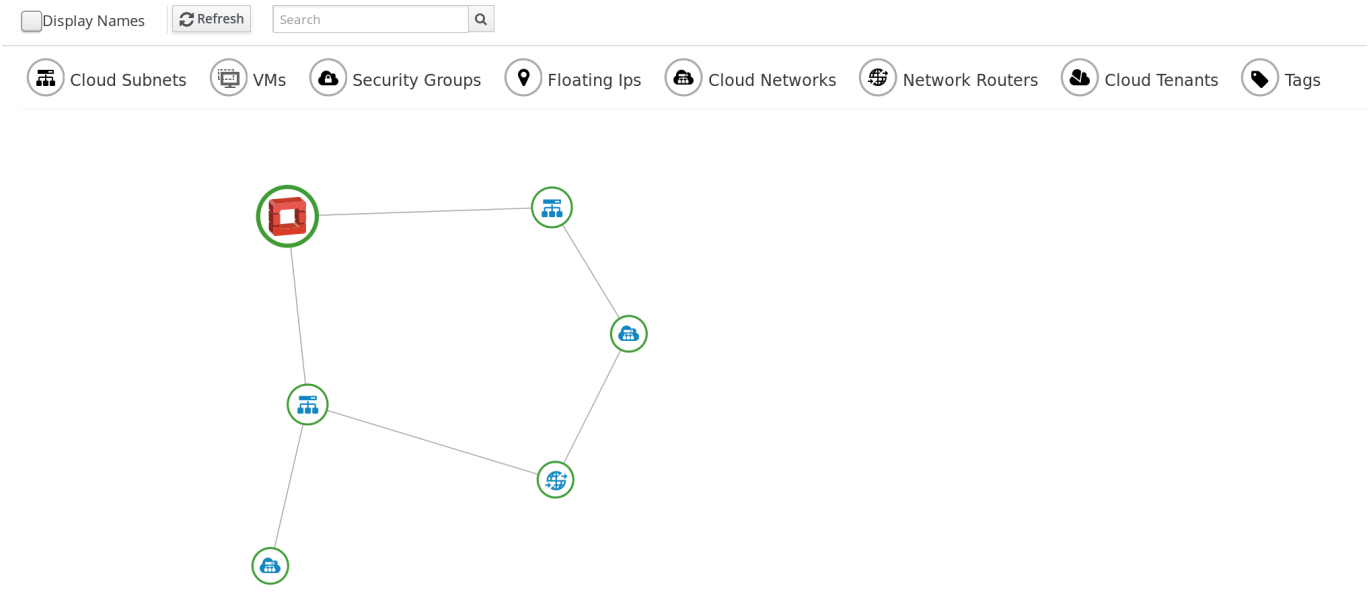
You can also assign policy profiles to network providers, or remove them. The method for doing so is similar to that of any normal policy profile.

Using the Topology Widget for Network Providers

The **Topology** widget is an interactive topology graph, showing the status and relationships between the different entities of the network providers that Infrastructure Automation has access to.

The topology graph includes cloud subnets, virtual machines, security groups, floating IP addresses, cloud networks, network routers, cloud tenants, and tags within the overall network provider environment.

Each entity in the graph displays a color indication of its status: green indicates an active entity, while red indicates inactivity or an issue.



1. Browse to menu: **Networks > Topology**.

2. Click the desired network provider for viewing the provider summary.

Alternatively, you can open the topology widget from the provider summary page by clicking **Topology** under **Overview**.

- Hovering over any individual graph element will display a summary of details for the individual element.
- Double-click an entity in the graph to browse to its summary page.
- Drag elements to reposition the graph.
- Click the symbols in the legend at the top of the graph to show or hide entities.
- Click the **Display Names** checkbox to show or hide entity names.
- Click **Refresh** to refresh the display of the network provider entities.
- Enter a search term in the **Search** box to locate an entity by full or partial name.

Containers Providers

A containers provider is a service that manages container resources, that can be added to the Infrastructure Automation appliance.

Infrastructure Automation can connect to OpenShift Container Platform containers providers and manage them similarly to infrastructure and cloud providers. This allows you to gain control over different aspects of your containers environment and answer questions such as:

- How many containers exist in my environment?
- Does a specific node have enough resources?
- How many distinct images are used?
- Which image registries are used?

When Infrastructure Automation connects to a container's environment, it collects information on different areas of the environment:

- Entities such as pods, nodes, or services.
- Basic relationships between the entities, for example: Which services are serving which pods?
- Advanced insight into relationships, for example: Which two different containers are using the same image?
- Additional information, such as events, projects, routes, and metrics.

You can manage policies for containers entities by adding tags. All containers entities except volumes can be tagged.

Note:

This chapter provides details on managing containers providers. For details on working with the resources within a container environment, see Container Entities in *Managing Infrastructure and Inventory*.

The Infrastructure Automation user interface uses virtual thumbnails to represent containers providers. Each thumbnail contains four quadrants by default, which display basic information about each provider:

1. Number of nodes
2. Container provider software
3. Power state
4. Authentication status

Icon	Description
	Validated: Valid authentication credentials have been added.
	Invalid: Authentication credentials are invalid.

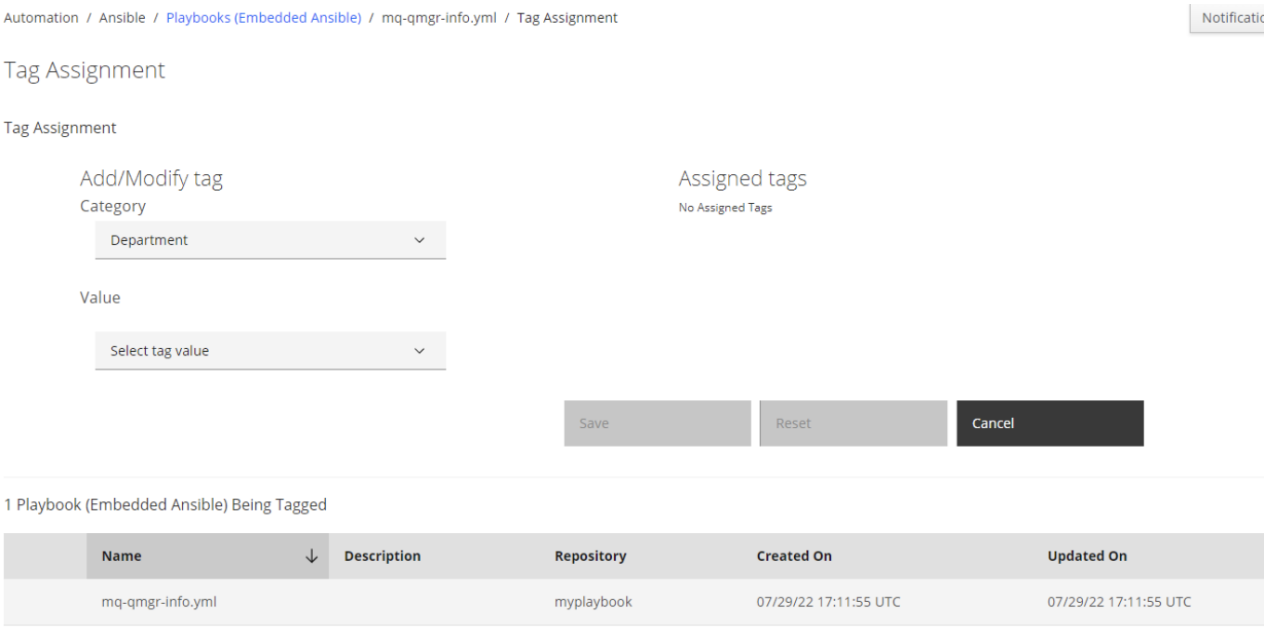
Icon	Description
?	Unknown: Authentication status is unknown or no credentials have been entered.

Containers provider authentication status

Tagging Containers Providers

Apply tags to all containers providers to categorize them together at the same time.

1. Navigate to **Compute > Containers > Providers**.
2. Select the checkboxes for the containers providers to tag.
3. Click (**Policy**), and then (**Edit Tags**).
4. Select a tag to assign from the drop-down menu.



5. Select a value to assign.
6. Click **Save**.

Removing Containers Providers

You may want to remove a containers provider from the VMDB if the provider is no longer in use.

1. Browse to menu: **Compute > Containers > Providers**.
2. Select the checkboxes for the containers providers to remove.
3. Click **Configuration**, and then (**Remove Containers Providers from Inventory**).
4. Click **OK**.

Editing a Containers Provider

Edit information about a provider such as the name, hostname, IP address or port, and credentials as required. If you have just upgraded your Infrastructure Automation environment from an older version, edit the provider to specify the authentication method the provider uses to connect to Infrastructure Automation.

1. Browse to menu: **Compute > Containers > Providers**.
2. Click the containers provider to edit.
3. Click **Configuration**, and then (**Edit Selected Containers Provider**).
4. Edit the **Name** if required.

   Note:

   The **Type** value is unchangeable.
5. Under **Endpoints** in the **Default** tab, edit the following as required:
   1. Select a **Security Protocol** method to specify how to authenticate the provider:
      - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.
      - **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.

        Note:

        You can obtain your OpenShift Container Platform provider's CA certificate for all endpoints (default, metrics, alerts) from `/etc/origin/master/ca.crt`. Paste the output (a block of text starting with `-----BEGIN CERTIFICATE-----`) into the **Trusted CA Certificates** field.
      - **SSL without validation**: Authenticate the provider insecurely (not recommended).
   2. Enter the **Hostname** (or IPv4 or IPv6 address) of the provider.

      Important:

      The **Hostname** must use a unique fully qualified domain name.
   3. Enter the **API Port** of the provider. The default port is `8443`.
   4. Enter a token for your provider in the **Token** box.

      Note:

      To obtain a token for your provider, run the `oc get secret` command on your provider; see Obtaining an OpenShift Container Platform Management Token.

      For example:

      # oc get secret –namespace management-infra management-admin-token-8ixxs –template='' | base64 –decode
   5. Click **Validate** to confirm that Infrastructure Automation can connect to the OpenShift Container Platform provider.
6. Under **Endpoints** in the **Metrics** tab, configure the following for gathering capacity and utilization metrics for Prometheus based on the selection:
   1. Select a **Security Protocol** method to specify how to authenticate the provider:
      - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.
      - **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.
      - **SSL without validation**: Authenticate the provider insecurely using SSL. (Not recommended)
   2. Enter the **Hostname** (or IPv4 or IPv6 address) of the provider.
   3. Enter the **API Port** if your provider uses a non-standard port for access. The default port is `443`.
   4. Click **Validate** to confirm that Infrastructure Automation can connect to the endpoint.
7. Under **Endpoints** in the **Alerts** tab, configure the following for Prometheus alerting from the cluster.
   - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.

- **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.
- **SSL without validation**: Authenticate the provider insecurely using SSL. (Not recommended)
    - Enter the **Hostname** (or IPv4 or IPv6 address) of the provider.
    - Enter the **API Port** if your provider uses a non-standard port for access. The default port is `443`.
    - Click **Validate** to confirm that Infrastructure Automation can connect to the endpoint.

8. Click **Save**.

Hiding Environment Variables for Containers Providers

You can restrict users from viewing container provider environment variables by configuring user roles.

This is useful as the environment variables panel can expose sensitive information, such as passwords, that you may not want certain users to view.

Note:

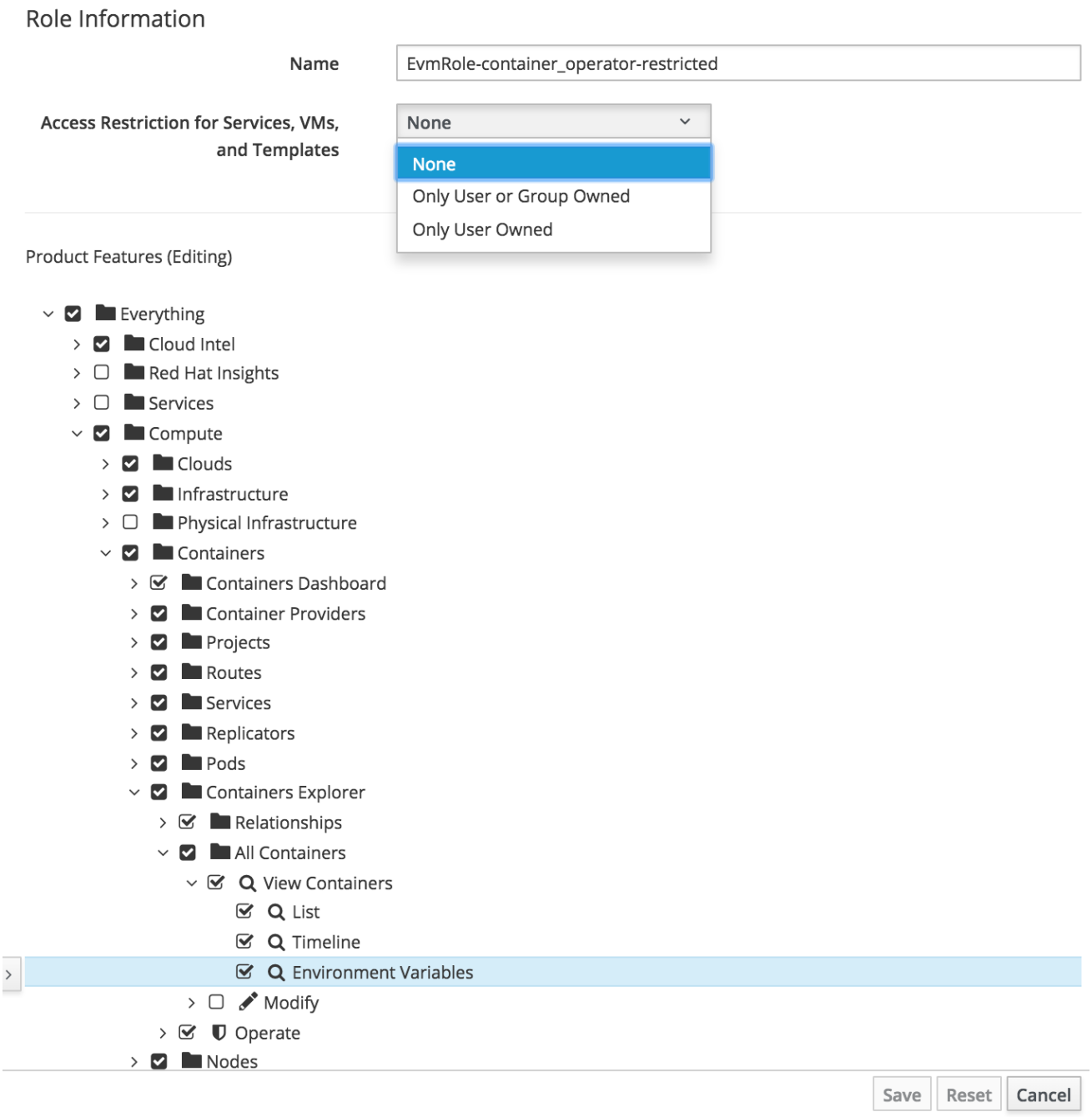
The default user roles in Infrastructure Automation are read-only. To customize a role's settings, create a new role or a copy of an existing role.

You can view role information and the product features the role can access (marked by a checkmark) by clicking on any role in **Access Control**. Expand the categories under **Product Features** to see further detail.

To configure user access to container environment variables:

1. Click (**Configuration**).
2. Click the **Access Control** accordion, then click **Roles**.
3. Select a existing custom role from the **Access Control Roles** list, and click (**Configuration**), then (**Edit the selected Role**).

   Alternatively, to create a new custom role, select a role from the **Access Control Roles** list, and click (**Configuration**), then (**Copy this Role to a new Role**).
4. Edit the name for the role if desired.
5. For **Access Restriction for Services, VMs, and Templates**, select if you want to limit users with this role to only see resources owned by the user or their group, owned by the user, or all resources (**None**).
6. Expand the **Product Features (Editing)** tree options to show **Everything > Compute > Containers > Containers Explorer > All Containers > View Containers**.
7. Clear the **Environment Variables** checkbox to restrict the user role from viewing container environment variables.



8. Click **Save**.

For more information about user roles, see Roles in *General Configuration*.

Viewing a Containers Provider's Timeline

View the timeline of events for instances registered to a containers provider.

1. Navigate to **Compute > Containers > Providers**.
2. Click the desired containers provider for viewing the timeline.
3. Click (**Monitoring**), and then (**Timelines**).
4. From **Options**, customize the period of time to display and the types of events to see.
   - Use **Show** to select regular Management Events or Policy Events.

- Use the **Interval** dropdown to select hourly or daily data points.
- Use **Date** to type the date for the timeline to display.
- If you select to view a daily timeline, use **Show** to set how many days back to go. The maximum history is 31 days.
- From the **Level** dropdown, select a **Summary** event, or a **Detail** list of events.
- The three **Event Groups** dropdowns allow you to select different groups of events to display. Each has its own color.

Click on an item for more detailed information.

- Azure Kubernetes Providers
- Red Hat OpenShift Providers
- IBM Cloud Kubernetes Service Providers
- Oracle Kubernetes Engine Providers
- VMware Tanzu Providers

Storage Managers

In Infrastructure Automation, a storage manager is a service providing storage resources that you can manage from a Infrastructure Automation appliance. This chapter describes the different types of storage managers used by Infrastructure Automation, and how they are added to Infrastructure Automation.

There are three types of storage managers currently available to Infrastructure Automation:

- Amazon Elastic Block Store Managers
- OpenStack Block Storage Managers
- OpenStack Object Storage Managers
- IBM Cloud Object Storage Managers

Appendix

Using a Self-Signed CA Certificate

Adding a self-signed Certificate Authority (CA) certificate for SSL authentication requires more configuration on OpenStack Platform and Microsoft System Center Virtual Machine Manager (SCVMM) providers.

Note: This procedure is not required for OpenShift Container Platform, or Red Hat Virtualization, which have the option to select **SSL trusting custom CA** as a **Security Protocol** in the user interface. These steps are needed only for providers without this option in the user interface.

Before adding the provider, configure the following:

1. Copy your provider's CA certificate in PEM format to **`/etc/pki/ca-trust/source/anchors/`** on your Infrastructure Automation appliance.
2. Update the trust settings on the appliance:

   ```
   # update-ca-trust
   ```

3. Restart the EVM processes on the server:

   ```
   # rake evm:restart
   ```

The CA certificate is added to the appliance, and you can add the provider to Infrastructure Automation.

- Overview
- Configuring ServiceNow Connection
- Managing Record Attributes
  - Create and Update (_patch) Record
    - Attribute
  - Get and Delete Record
  - Get All Records

- Use Cases
  - Provisioning a Virtual Machine from a Template
  - Virtual Machine Retirement
  - Virtual Machine Reconfiguration (VMware Only)

Overview

Infrastructure Automation integration with ServiceNow enables authentication with an existing ServiceNow database and **add/amend** items in the ServiceNow database during state machine processing, such as the virtual machine provisioning state machine. Note that information in this guide assumes you have credentials and access to a ServiceNow database instance.

The following new namespace and class delivers support for the management of **ServiceNow Configuration Management Database** (CMDB) records using **ServiceNow's RESTful** web service.

`/Integration/ServiceNow/CMDB`

You can manage records in the **CMDB_CI_SERVER** table, including **`create`**, **`update`**, and/or **`delete`**. The following methods are included:

Method	Action
create	Create record in specified ServiceNow table.
delete	Delete record in specified ServiceNow table.
get	Get record from specified ServiceNow table and list its attributes.
get_all	Get all records in ServiceNow and list attributes.
update	Get specified record, update required attributes and post updated record.
update_patch	Post required attributes to specified record.

Note:

Configuration item (CI) and record are used interchangeably and refer to items in a ServiceNow database table.

Configuring ServiceNow Connection

Configure the connection to the ServiceNow database by specifying the credentials in the **CMDB schema** or instances within.

The following methods are included:

snow_server	ServiceNow database IP address or resolvable hostname.
snow_user	ServiceNow user account with the necessary permissions.
snow_password	Associated user account password.

The table name cannot be changed unless there is a specific requirement to manage records elsewhere. Entries in this table appear in the **Configuration > Base Items > Servers** menu in the ServiceNow web user interface.

table_name	ServiceNow cmdb_ci_server database table.

Managing Record Attributes

You can specify any attribute via the **URI** or **CMDB** class instance. If neither exists, the value is determined from the Infrastructure Automation **VM** or **miq_provision** objects.

The attributes can be reduced or extended as required by amending the Ruby methods.

Note:

Some attributes are not free-text (variable) fields, that is, they must be specific values. For example, the **`vendor`** value must already exist in the ServiceNow Vendor table beforehand.

Create and Update (_patch) Record

You can use the following attributes when creating and/or updating a record.

Attribute

Attribute	Value
virtual	Boolean true or false, set to true.

name VMs	Virtual infrastructure name.
short_description	Infrastructure Automation virtual machine GUID.
host_name	The virtual machine's operating system hostname.
cpu_count	The virtual machine's CPU count.
ram	The virtual machine's memory.
vendor	The virtual machine's hardware vendor (provider).
sys_id [1]	ServiceNow record unique system ID.

Get and Delete Record

You can use the following attribute for getting or deleting a record.

Attribute	Value
sys_id	ServiceNow record unique system ID.

Get All Records

There are no attributes required to get all records. This method gets all records in the specified **ServiceNow table** and writes their attributes to `automation.log`.

1. sys_id attribute is not required during `create`. Its value is returned from the `create` request and the Infrastructure Automation object custom attribute **servicenow_sys_id** is created and updated.

Use Cases

The following examples show how ServiceNow can be integrated with automation workflows.

Provisioning a Virtual Machine from a Template

The Cloud and Infrastructure **Provision VM from Template** State Machines contain **RegisterCMDB** and **ActivateCMDB** states.

To create a new ServiceNow record during virtual machine provisioning, amend the **ActiveCMDB** state to call the `create` method, for example:

`/Integration/ServiceNow/CMDB/create`

ActivateCMDB	/Integration/ServiceNow/CMDB/create

Virtual Machine Retirement

The Cloud and Infrastructure default **Retirement** State Machines contain the **DeactivateCMDB** state.

To update a virtual machine's ServiceNow record during virtual machine retirement, amend the **DeactivateCMDB** state to call the `update_patch` method, for example:

`Integration/ServiceNow/CMDB/update_patch?description=VM%20${/#vm.guid}%20retired%20from%20{productname_short}`

DeactivateCMDB	/Integration/ServiceNow /CMDB/update_patch?description=VM%20${/#vm.guid}%20retired%20from%20CloudForms

Virtual Machine Reconfiguration (VMware Only)

Create a new **System Event** instance to update the ServiceNow record after a virtual machine reconfiguration request has been approved and completed.

Create a new **/System/Event/ReconfigVM_Task_Complete** instance with a relationship value:

`/Integration/ServiceNow/CMDB/update_patch`

rel1	/Integration/ServiceNow/CMDB/update_patch

Change risk integration for Infrastructure Automation

You can enable an integration connection for Infrastructure Automation to gain the advantage of change risk assessments. With this integration, your IT operators can use data from ServiceNow to assess the risk of a code deployment based on prior incidents, estimate the chances of a successful update, identify risks, build logs, apply security policies and more.

The Change Risk module ingests un-structured data from ServiceNow, and analyzes the data to look for similar tickets and to understand how incidents and failures might have occurred. The module can then provide an assessment of the overall risk of suggested changes. With an integration with ServiceNow for change risk, Infrastructure Automation allows for the automation of the delivery of infrastructure services which integrates with ServiceNow. Since Infrastructure Automation is an end-to-end infrastructure provisioning and management capability, Infrastructure Automation can complete ongoing checks against configurations after infrastructure is provisioned to ensure systems are running as prescribed and adhere to system-wide compliance policies. Through the integration, updates can be made to the ServiceNow CMDB or to create change requests.

With the integration in place, the following flow demonstrates the possible interactions between Infrastructure Automation and the Change Risk components of IBM Cloud Pak for AIOps and ServiceNow:

Scenario: A change is proposed to add additional disk space to a virtual machine.

Interaction Flow:

1. The Operator invokes a day-2 operation in Infrastructure Automation to change the server configuration to create and attach the new disk.
2. The day-2 operation creates a Change Request ticket in ServiceNow, populating it with details on the requested change (virtual machine ID, new disk size, and more).
3. The Change Risk component monitors the ServiceNow tickets. Using a pre-trained model to analyzing similar tickets, IBM Cloud Pak for AIOps can classify the change as a high or low risk change and add a confidence level of the prediction onto the ServiceNow ticket.
4. The ServiceNow operator approves the change request based on the confidence level, and marks it ready for implementation.
5. The day-2 operation in Infrastructure Automation waits for the ServiceNow Change Request to be approved before proceeding.
6. After observing the change in the state of the ServiceNow Change Request, Infrastructure Automation carries out the provisioning change to add the disk to the virtual machine.
7. After completing the provisioning action, Infrastructure Automation updates the ServiceNow Change Request to mark it as complete. The request can be reviewed and closed by a ServiceNow operator.
8. Infrastructure Automation policies are applied against the configuration and alerts are raised if server configuration isn't in compliance. Both server VM parameters as well as application middleware packages can be checked for compliance against defined policies.

For more information abouth the steps to configure the AI algorithm to determine change risk, see Configuring change risk.

Turbonomic connection

You can enable a cloud connection between Infrastructure Automation and Turbonomic to integrate the systems. You can use Turbonomic for capacity planning to help identify usage patterns, trends and the optimization of application resource usage to help you reduce waste and control costs while meeting application service level agreements (SLA). Turbonomic also provides insights and recommended actions based on multiple sources of data. With this integration, Turbonomic delivers insights directly to the Infrastructure Automation service management so that your DevOps personnel can act on the insights when needed.

With an integration connection set up between Infrastructure Automation and Turbonomic, you can take advantage of the following capabilities:

- You can use Turbonomic to monitor your cloud environment. This includes monitoring infrastructure provisioned through Infrastructure Automation.
- Infrastructure Automation can run multiple workflow steps by using Terraform and Ansible, which can include steps to update ServiceNow, issue notifications, and configuring the application inside the virtual machine (VM).
- Infrastructure Automation can run steps for the workflow, such as to resize the VMs and update the ServiceNow configuration management database (CMDB) as dictated by the parameters of the action recommended by Turbonomic.
- Turbonomic can monitor and identify VMs that can be resized to save resources while keeping the application SLA.
- Turbonomic can associate the VM to the Infrastructure Automation service that provisioned it by using ingested tags from the provisioned infrastructure.
- Turbonomic can invoke Infrastructure Automation REST API to resize the VM in a particular service instance by using a custom action script that is registered to Turbonomic and defined in a policy specific to the service.
- Turbonomic can view the updated VM, measure the performance, and ensure that the application SLA is being met with the changed configuration.

For Infrastructure Automation to access your Turbonomic environment, you must provide the value of the following Turbonomic cloud connection parameters to enable the integration connection:

- **Turbonomic server URL** - The authentication URL, which is used for the identification purposes. Use the format `http://<IP_ADDRESS>`.
- **User Name** - The user name for logging into the Turbonomic server.
- **Password** - The password for the provided user name.

For more information about these values and how to configure them, see Target Configuration.

Enabling or disabling IBM Business Process Manager

Post the installation of Managed services, you can enable or disable Business Process Manager from the IBM Cloud Pak for AIOps.

Procedure

Complete these steps from your OpenShift console.

1. Click **Operators** > **Installed Operators**.
2. Select **`cp4aiops`** namespace from the **Project** drop-down list.
3. Select the operator **`IBM Management Infrastructure managed services`**.
4. Select the tab Managed services.
5. Edit the Managed services instance by selecting **Edit ManageService** from the overflow menu.
6. Search **`camBpmProvider`** in YAML and change the following values:
   - Change **`replicaCount`** to 1.
   - Edit **`bpmEndpoint`** with the BPM endpoint value.
   - Edit **`bpmUserName`** with the user name value.
   - Edit **`bpmUserPwd`** with the password.

     These are the sample values for BPM provider:

     ```
     camBpmProvider:
       bpmEndpoint: 'https://9.9.9.9:9443/'
       bpmUserName: 'admin'
       bpmUserPwd: 'passw0rd'
       replicaCount: 1
     ```
7. Save the YAML file.

Note: The supported version of IBM Business Process Manager is 8.5.7.

Enabling or disabling Ansible Integration

Post the installation of Managed services, you can enable or disable Ansible integration from the IBM Cloud Pak for AIOps.

Procedure

Complete these steps from your OpenShift console.

1. Click **Operators** > **Installed Operators**.
2. Select **`cp4aiops`** namespace from the **Project** drop-down list.
3. Select the operator **`IBM Management Infrastructure managed services`**.
4. Select the tab Managed services`.
5. Edit the Managed services instance by selecting **Edit ManageService** from the overflow menu.
6. Search **`camAnsibleProvider`** in YAML and change the **`replicaCount`** to 1.
7. Save the YAML file.

Creating IBM Infrastructure management connections

You can send event data from Infrastructure management to IBM Cloud Pak for AIOps through an Infrastructure management connection.

- Before you begin
- Creating Infrastructure management connections
- Editing Infrastructure management connections
- Deleting Infrastructure management connections

Before you begin

Before you configure an Infrastructure management connection, ensure that the following prerequisites are met:

- You have a working Infrastructure management cluster. Infrastructure management must be installed on an Red Hat OpenShift Container Platform cluster. The connection cannot be created for an Infrastructure management appliance that is running on a virtual machine (VM).
- and that event syndication is enabled within Infrastructure management.

  For information on how to enable event syndication, see *Enabling Events Syndication* within the topic General Configuration

Note: Infrastructure management and Cloud Pak for AIOps can be located in the same cluster or in different clusters.

Creating Infrastructure management connections

To create an Infrastructure Management connection, complete the following steps:

1. Log in to the IBM Cloud Pak Automation console.
2. Expand the navigation menu (four horizontal bars), then click **Define** > **Data and tool connections**.
3. On the **Data and tool connections** page, click **Add connection**.
4. From the list of available connections, find and click the Infrastructure Management tile.

   Note: If you do not immediately see the connection that you want to create, you can filter the tiles by type of connection. Select the **Events** type from the **Category** list.
5. On the side panel, review the instructions. When ready to continue, click **Connect**.
6. On the Add connection page, define the general connection details:

   Name: The display name for the connection.

   Description (optional): An optional description for the connection.
7. Click Done.

   Important: After creating the connection, you are prompted to copy or download a boostrap command for running a remote deployment script. The script is generated when you create the connection in IBM Cloud Pak for AIOps. This script is required for establish the connection from Infrastructure management to IBM Cloud Pak for AIOps.
8. Copy or download the bootstrap command for the deployment script.

   You can click the **Copy to clipboard** button to copy the command, or click **Download as a sh-file** to download the command as a `bootstrap.sh` file.

   Note: If you do not obtain the command now, you can copy or download the command later by completing the following steps:

   1. Log in to IBM Cloud Pak Automation console.
   2. Expand the navigation menu (four horizontal bars), then click **Define** > **Data and tool connections**.
   3. Click the **IBM Infrastructure Management** connection type on the *Manage connections* tab of the *Data and tool connections* page.
   4. On the IBM Infrastructure Management page, click the **Download** link in the **Remote deployment script** column for the connection.
   5. Either copy or download the bootstrap command for the deployment script.
9. Run the remote deployment script on your Infrastructure management cluster.

   1. Get the OpenShift CLI `oc login` command from the cluster where Infrastructure management is installed.
   2. From a command line, log in to the Infrastructure management cluster with the `oc login` command.
   3. Switch to the project (namespace) where Infrastructure management is installed.
   4. Run the deployment script as a `sh` (script file), such as the downloadable `bootstap.sh` file.

Editing Infrastructure management connections

After you create a connection, you can edit the connection to update the name or description. Complete the following steps to edit the connection:

1. Log in to IBM Cloud Pak Automation console.

2. Expand the navigation menu (four horizontal bars), then click **Define** > **Data and tool connections**.
3. Click the **IBM Infrastructure Management** connection type on the *Manage connections* tab of the *Data and tool connections* page.
4. On the IBM Infrastructure Management page, click the name of the connection that you want to edit. Alternatively, you can click the options menu (three vertical dots) for the connection and click **Edit**. The connection configuration opens.
5. Edit your connection as required.
6. Click **Save** when you are done editing.

Deleting Infrastructure management connections

If you no longer need your IBM Infrastructure management connection and want to not only disable it, but delete it entirely, you can delete the connection from the console.

To delete a connection, you need to complete the following steps for IBM Cloud Pak for AIOps. Then, you need to complete steps on your Infrastructure management to remove the connection deployment. The deployment on the cluster is not removed when you delete the connection in the IBM Cloud Pak Automation console for IBM Cloud Pak for AIOps.

1. Log in to IBM Cloud Pak Automation console.
2. Expand the navigation menu (four horizontal bars), then click **Define** > **Data and tool connections**.
3. Click the **IBM Infrastructure Management** connection type on the *Manage connections* tab of the *Data and tool connections* page.
4. On the IBM Infrastructure Management connections page, click the options menu (three vertical dots) for the connection that you want to delete and click **Delete**.
5. Enter the name of the connection to confirm that you want to delete your connection. Then, click **Delete**. Your connection is deleted.
6. Delete the connection deployment on your Infrastructure management cluster. To delete the deployment, you must manually run OpenShift CLI (`oc`) commands against the cluster.
    1. From a command line, use the `oc login` command to log in to your Infrastructure management cluster.
    2. Switch to the project (namespace) where Infrastructure management is installed.

       ```
       oc project <project>
       ```

       Where `<project>` is the project (namespace) where Infrastructure management is installed.
    3. Delete the `grpc-bridge-secret` secret.

       ```
       oc delete secret grpc-bridge-secret
       ```
    4. Delete the connection deployment

       ```
       oc delete deploy aiops-connector
       ```

       The deployment is now deleted from your Infrastructure management cluster.

Migrating infrastructure data to Infrastructure Automation

You can migrate infrastructure data, such as services data and appliances, to Infrastructure Automation from other products and services, such as IBM Cloud Pak for Multicloud Management, IBM Cloud Private, and Red Hat CloudForms.

- Migrating IBM Cloud Private Managed services
- Upgrading Red Hat CloudForms and IBM Cloud Pak for Multicloud Management appliances

Migrating IBM Cloud Private Managed services to Infrastructure Automation

This topic describes the process of migrating Managed services from IBM Cloud Private to Infrastructure Automation.

Most of the Managed services pods are stateless, except MongoDB pod. The MongoDB pod stores state in external volumes, that is, persistent volumes. You must do the migration for the following resources:

- Logs

  The logs from different MongoDB pods in IBM Cloud Private are stored in persistent volumes, but in Infrastructure Automation, the pods don't use persistent volumes to store logs. The Infrastructure Automation uses **`system.out`** as default logging mechanism. It is recommended to use an external log aggregator like Kibana for the IBM Cloud Private customers.

- Terraform binaries

  In IBM Cloud Private, the custom terraform binaries are stored in persistent volumes whereas in Infrastructure Automation, the custom terraform binaries are stored in MongoDB. If in case, you have a custom terraform binaries or want to use binary version's that is not shipped anymore as part of the Infrastructure Automation release, then you can upload using Terraform file APIs.

- MongoDB

  The MongoDB pod contains IBM Cloud Private state. This state needs to be backed up and restored in Infrastructure Automation. For more information, see Backing up MongoDB in IBM Cloud Private and Restoring MongoDB.

Backing up MongoDB in IBM Cloud Private

1. Stop the running pods by setting the replica count to 0 and ensure that nothing is written to the MongoDB pod or scale down the MongoDB pod to 0.
2. Start the MongoDB pod and set replica of MongoDB pod count to 1.
3. Switch to the service namespace.

   ```
   kubectl config set-context $(kubectl config current-context) --namespace=services
   ```

4. Create a new MongoDB pod using the **`mongopod.yaml`** file and ensure to use the same labels as other pods.

   ```
   kind: Pod
   apiVersion: v1
   metadata:
     name: cam-mongo-backup
     labels:
       app.kubernetes.io/instance: cam
       app.kubernetes.io/managed-by: Helm
       app.kubernetes.io/name: cam-manageservice
       certmanager.k8s.io/time-restarted: 2022-2-23.1227
       helm.sh/chart: manageservice-4.2.0
       name: cam-mongo-backup
       release: cam
     annotations:
       openshift.io/scc: anyuid
   spec:
     containers:
       - name: backup-mongo
         image: mongo:4.0.16
   ```

   Note: Get the mongo version of the targeted cluster by executing inside the mongo pod and running **`mongo --version`**. Use the same tag of mongo image while creating the backup mongo pod.

   Run **`kubectl apply -f mongopod.yaml`** to create the **`cam-mongo-backup`** pod and wait for the pod to get started.

   If you encounter with an error message like **`no matching repositories in ClusterImagePolicy`** while applying the **`kubectl`**, then edit the cluster policy and add image policy for the MongoDB pod as following:

   ```
   kubectl  edit ClusterImagePolicy image-policy

   -name:docker.io/mongo:*
   ```

5. Get the IP address of MongoDB pod.

   ```
   MONGO_IP=$(kubectl get -n management-infrastructure-management svc/cam-mongo --no-headers | awk '{print $3}')
   ```

6. Get password to connect to the MongoDB pod.

   ```
   MONGO_PASSWORD=$(kubectl get -n management-infrastructure-management secret cam-secure-values-secret -o yaml | grep mongoDbPassword: | awk '{print $2}' | base64 --decode)
   ```

7. Get the username and port by describing **`cam-mongo`** pod.

   ```
   kubectl describe pod cam-mongo
   ```

8. Execute the **`mongodump`** from **`cam-mongo-backup`** pod to create a zip file.

```
kubectl -it exec cam-mongo-backup -- bash -c "mongodump --ssl --sslAllowInvalidCertificates --uri
mongodb://MONGODB_USERNAME:$MONGO_PASSWORD@$MONGO_IP:27017/cam?ssl=true --archive=/tmp/mongo_backup.gz --gzip"
```

Note: Verify whether the tar file is present at **/tmp** location in the pod.

9. Run the following command to copy the back up archive to your local set up, for example, the **/tmp/dbbackup** directory

```
kubectl cp cam-mongo-backup:/tmp/mongo_backup.gz /tmp/dbbackup
```

10. Delete **cam-mongo-backup** pod.

```
kubectl delete pod cam-mongo-backup
```

11. Restore the replica count for the pods that you scaled down in a previous step.

Restoring MongoDB in Infrastructure Automation

The MongoDB contains Managed services state and to restore this state, do the following.

1. Stop the running pods by setting the replica count to 0 and ensure that nothing is written to the MongoDB pod or scale down the MongoDB pod to 0.
2. Start the MongoDB pod and set replica of MongoDB pod count to 1.
3. Switch to the **cp4aiops** namespace.

```
oc project cp4aiops
```

4. Create a new MongoDB pod using the **mongopod.yaml** file and ensure to use the same labels as other pods:

```
kind: Pod
apiVersion: v1
metadata:
  name: cam-mongo-backup
  labels:
    app.kubernetes.io/instance: cam
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: cam-manageservice
    certmanager.k8s.io/time-restarted: 2022-2-23.1227
    helm.sh/chart: manageservice-4.2.0
    name: cam-mongo-backup
    release: cam
  annotations:
    openshift.io/scc: anyuid
spec:
  containers:
    - name: backup-mongo
      image: mongo:4.0.16
```

Note: Get the mongo version of the targeted cluster by executing inside the mongo pod and running **mongo --version**. Use the same tag of mongo image while creating the backup mongo pod.

Run **oc apply -f mongopod.yaml** to create the **cam-mongo-backup** pod and wait for the pod to get started.

5. Copy the desired back up archive to restore into the **cam-mongo-backup** container.

```
oc rsync /tmp/dbbackup/ cam-mongo-backup:/tmp/
```

6. Get the IP address of MongoDB pod.

```
MONGO_IP=$(kubectl get -n cp4aiops svc/cam-mongo --no-headers | awk '{print $3}')
```

7. Get password to connect to the MongoDB pod.

```
MONGO_PASSWORD=$(kubectl get -n cp4aiops secret cam-secure-values-secret -o yaml | grep mongoDbPassword:
| awk '{print $2}' | base64 --decode)
```

8. Get the username and port by describing **cam-mongo pod**.

```
oc describe pod cam-momgo
```

9. Restore the data by running the following command:

```
oc -it exec cam-mongo-backup -- bash -c "mongorestore --ssl --sslAllowInvalidCertificates --uri
mongodb://camuser:$MONGO_PASSWORD@$MONGO_IP:27017/cam?ssl=true --archive=/tmp/mongo_backup.gz --gzip --
drop"
```

10. Delete **cam-mongo-backup** pod.

```
oc delete pod cam-mongo-backup
```

11. Restore the replica count for the pods that you scaled down in a previous step.
12. For the services that are not Globally accessible, you need to create the namespaces if they do not already exist. Create the namespaces in the cluster where you are restoring data so that the services can be accessed.

Migrating Red Hat CloudForms and IBM Cloud Pak for Multicloud Management appliances to Infrastructure Automation

This topic describes the process of doing a full appliance replacement upgrade when a new release of Infrastructure Automation – Infrastructure Management is available.

The following migration paths are supported:

- IBM Cloud Pak for Multicloud Management 2.3.x - Infrastructure management to IBM Cloud Pak for AIOps 4.2.1 - Infrastructure Automation
- Red Hat CloudForms 4.7 through 5.0 to IBM Cloud Pak for AIOps 4.2.1 - Infrastructure Automation

Ordering migrations

You must complete all migrations of the databases before you configure replication and high availability. The reason is because you deploy new appliances for migrating to Infrastructure management.

Important: Do not run `evmserverd` if a database is in a partially migrated state.

Modifying the PostgreSQL configuration

This procedure modifies the strategy that is used for updating the PostgreSQL configuration. Previously, changes were made directly to the `postgresql.conf` file in the data directory. After you update the new configuration, changes will be made by editing a separate file in the `/etc/manageiq/postgresql.conf.d` directory. This modification allows future PostgreSQL configuration changes to be made more easily.

Note: The `01_miq_overrides.conf` file in the `/etc/manageiq/postgresql.conf.d` directory is reserved for application overrides and is re-created on application updates. **Important:** This file must not be edited. More files are read by the PostgreSQL server process according to the documentation here: PostgreSQL - Managing Configuration File Contents

1. Back up the `postgresql.conf` file so that you have any configuration changes:

   ```
   cp $APPLIANCE_PG_DATA/postgresql.conf <file_location>
   ```

2. Copy the updated `postgresql.conf` file to the `pg_data` directory:

   ```
   cp $APPLIANCE_SOURCE_DIRECTORY/TEMPLATE/var/opt/rh/rh-postgresql95/lib/pgsql/data/postgresql.conf
   $APPLIANCE_PG_DATA/postgresql.conf
   ```

3. Optional: If you updated the `postgresql.conf` file before this procedure, create a new `02_user_overrides.conf` file, and add any modifications that you have from the original `postgresql.conf` file. Example layout of the `02_user_overrides.conf` file:

   ```
   #---------------------------------------------
   # WRITE AHEAD LOG
   #---------------------------------------------

   wal_level = 'logical'
   wal_log_hints = on
   wal_buffers = 16MB
   min_wal_size = 1GB
   max_wal_size = 2GB
   checkpoint_completion_target = 0.9
   ```

   After the `02_user_overrides.conf` file is modified and saved, restart the postgresql service:

   ```
   systemctl restart $APPLIANCE_PG_SERVICE
   ```

All `postgresql.conf` customizations are now made to files in the `/etc/manageiq/postgresql.conf.d` directory due to the following `include_dir` line in the default configuration. The settings in these new files override ones in the `postgresql.conf` file.

```
include_dir = '/etc/manageiq/postgresql.conf.d`
```

Migrating to the Infrastructure Management appliance

Overview

This procedure describes the process of migrating to the Infrastructure management appliance. This procedure does not necessarily include migration of all possible customer modifications, so it is recommended that you fully test any modifications before you migrate your environment.

Important:

- Read through all of the steps in this procedure before you begin the migration process.
- Infrastructure management appliances require 12 GB memory. However, before you migrate your appliances, adjust resources in your environment to avoid performance issues.

You can classify the migration into three groups of appliances:

- VMDB appliance - An appliance with running workers that also hosts its own database that other appliances can also connect to.
- Non-VMDB appliance - An appliance with running workers that does not host a database. The appliance is connected to an external database.
- Dedicated database appliance - A Infrastructure management appliance or non-Infrastructure management virtual machine with no running workers on it: the appliance contains only a database for other appliances to connect to.

Migration workflow summary

In summary, the migration process follows this workflow:

1. Prepare appliances:

   Note: This step applies only for a multi-region Infrastructure management environment. Remove all the subscriptions from the global region before you migrate the global database since the subscriptions must be dropped and re-created.

2. Back up the database of your appliance.

3. Migrate appliances:

   1. Provision new appliances to replace each appliance.

   2. On the database and nondatabase appliances, copy the GUID file from the version of the same appliance to the same location.

   3. On the database and nondatabase appliances, use the appliance console menu to stop the evmserverd service, manually remove the pre-configured database and key, and fetch the v2_key from an appliance.

   4. Copy the appliance database backup file and restore the database on each VMDB and dedicated database appliance.

   5. Load the new version of the pglogical library on the VMDB and dedicated database appliances.

      Note: This step applies only for a multi-region Infrastructure management environment.

   6. Migrate the VMDB appliance databases and update the Automate Model.

   7. Restart **evmserverd** on the VMDB appliances.

4. Configure replication after the migration process is complete and appliances are running again.

Backing up current appliances

These steps do not affect the operations of your appliance infrastructure. However, they help ensure that you are able to roll back if required and replicate the network settings.

1. Find the size of the databases by using the following command. Then, ensure that your volume has enough space for the backup file.

   ```
   psql -d vmdb_production -c "SELECT pg_size_pretty(pg_database_size('vmdb_production'));"
   ```

2. If it's a multi-region environment, ensure you removed all subscriptions from the global database appliance.

3. Stop the **evmserverd** process.

   ```
   systemctl stop evmserverd
   ```

4. Back up the databases of your appliances in the root directory. Take a snapshot if possible.

   ```
   pg_dumpall -c --if-exists | gzip > vmdb.pg.gz
   ```

   Notes:

   - This command creates a compressed text-based backup. **pg_dumpall** is a utility for writing out ("dumping") all PostgreSQL databases of a cluster into one script file. For more information, see pg_dumpall.
   - When you run a **pg_dumpall**, the **pg_replication_slots** is not copied over causing the slots to become inactive. Also, the size of the global database appliance is reduced allowing for quicker backup and restore.

- Even a compressed database backup can be large so consider storing it on a file system with adequate space such as /var/tmp. Review your file system available space by using the df command.
- You must also provide backup for configuration files such as **/var/www/miq/vmdb/certs/v2_key**, **/var/www/miq/vmdb/config/database.yml**, and **/var/www/miq/vmdb/GUID**.

5. Back up the following files and take note of the appliance that each file comes from. These files are used to replace the old appliances in the database and for disaster recovery.

   - /var/www/miq/vmdb/GUID

6. During the upgrade, the iptables configuration file **/etc/sysconfig/iptables** is removed. If you changed the iptables configuration from the default, run the **iptables --list -n** command to see the current configuration. Run the following command to back up the iptables configuration:

   ```
   iptables-save > /etc/iptables.conf
   ```

   You can restore your iptables configuration file with the following command:

   ```
   iptables-restore < /etc/iptables.conf
   ```

   You can also add this command to **/etc/rc.local** to reload the rules at every restart.

Steps to follow on newly deployed Infrastructure management 4.2.1 Appliances

Complete the following steps on your appliances as specified to migrate to Infrastructure management.

Note: Some steps are run on certain appliances. Ensure you wait for each command to finish before you move to the next step.

1. Provision and start the new appliances.
2. On the VMDB and non-VMDB appliances, run the following commands as the root user:
   1. Copy the GUID file from the appliance.

      ```
      scp root@$IP_of_old_appliance:/var/www/miq/vmdb/GUID /var/www/miq/vmdb/
      ```

      Notes:

      - Complete these steps on VMDB appliances first. The non-VMDB appliances don't have a database to connect to if they're done first.
      - Complete this step only if you want to keep the same appliance/miq_servers records in the database and inherit that appliance's existing settings. If you plan on deploying new appliances with new IP addresses and hostname and not inherit the settings, you might prefer creating new records by not copying the GUID.

   2. Stop the evmserverd process.

      ```
      systemctl stop evmserverd
      ```

   3. Remove the previously generated encryption key.

      ```
      rm /var/www/miq/vmdb/certs/v2_key
      ```

   4. Copy the default database configuration file, ovewriting the previously generated one that used the deleted encryption key.

      ```
      scp root@<IP_of_old_appliance>:/var/www/miq/vmdb/config/database.yml
      /var/www/miq/vmdb/config/database.yml
      ```

   5. Drop the previously generated database.

      ```
      dropdb vmdb_production
      ```

   6. For VMDB appliances, configure the application, fetch the key from remote system, and Reset Configured Database.
   7. For VMDB appliances, select the same region number and database password as the old installation. You can find the region number that is used for an installation that is listed as **Database/Region:** in the **appliance_console** summary information.
   8. For non-VMDB appliances, configure the application by using the join region in external database option. Note as part of the database configuration step, fetch the **v2_key** from the appliances.
   9. Allow the appliance to start.

3. On the new VMDB and new dedicated database appliances:
   1. Stop the evmserverd process.

      ```
      systemctl stop evmserverd
      ```

   2. Drop the default database.

      ```
      dropdb vmdb_production
      ```

3. Copy the database dump from the appliance. Update the following location to reflect the source file location of the database dump and copy it to a location with enough available space.

   ```
   scp root@$IP_of_old_appliance:/var/tmp/vmdb.pg.gz /var/tmp
   ```

4. Decompress and restore the database dump backup file.

   ```
   gunzip < vmdb.pg.gz | psql postgres
   ```

5. Change to the vmdb directory.

   ```
   cd /var/www/miq/vmdb/
   ```

6. Run the command appropriate to your environment to migrate everything in the database to work with the latest configuration. If you run into an error, contact IBM Support.

   ```
   rake db:migrate
   ```

4. Update the Automate Model to the latest version. This resets the ManageIQ and IBM domains (base domains) to a new and upgraded version. In a single (stand-alone database) or replicated environment, run this command on each VMDB appliance.

   ```
   rake evm:automate:reset
   ```

5. Start the evmserverd process. In a single (stand-alone database) or replicated environment, run this command on each VMDB appliance:

   ```
   systemctl start evmserverd
   ```

The migration process is now complete. Once appliances are running again, you can configure database replication in your environment. For more information, see Configuring Database Replication and Centralized Administration in the General Configuration guide.

Infrastructure Automation backup and restore

Learn how to back up and restore IBM Cloud Pak® for AIOps Infrastructure Automation, such as for recovering from a disaster like a complete data center outage.

- Overview
- Planning
- Backup process
- Restore process
- Components

Overview

IBM Cloud Pak for AIOps backup and restore is based on the storage and database backup of critical datastores, cluster resources, and component data and settings. With the backup and restore feature, you can recover your services to a previous point in time backup to recover from data corruption, system failures, or user errors.

You can also use the backup and restore feature as a means of copying data from one cluster to another, or from one environment into another, such as for disaster recovery purposes.

If you are using backup data to restore environments on a new cluster in another data center, your recover time objective (RTO) and recovery point objective (RPO) must be sufficiently large for this usage.

IBM Cloud Pak for AIOps uses the *OpenShift APIs for Data Protection* (OADP) and the open source tool Velero to backup data to object storage. In the case of a failure, the APIs and Velero are used to restore data from the backups.

For more information about these tools, see:

- OpenShift APIs for Data Protection (OADP) features
- Velero documentation

Planning for backup and restore

When you are planning your IBM Cloud Pak for AIOps backup and restore strategy, consider what hardware requirements are needed for backing up your environment:

- You need to install the backup and restore tools in the Red Hat OpenShift Container Platform cluster. These tools include the OpenShift APIs for Data Protection (OADP), Velero, and Restic. You need to configure the OpenShift APIs and Velero with the appropriate object storage for storing your backups. This object storage must support RWX (read-write-execute) mode.
- You can backup both smaller starter and larger production sized cluster deployments.

- You can use backed up data to restore data to the existing cluster where the data was backed up, or to a new cluster.
- A restored cluster should have the same size and high availability level as the original cluster.
- Data is copied and stored in object Storage as part of the backup process. During a restoration, the same data is fetched and restored.

Notes:

- The backup and restore processes can take time. Using a backup to restore data or configurations is most useful when your recover time objective (RTO) and recovery point objectives (RPO) are met.
- As your data grows, the size of your backup storage might need to grow.
- A backup and restore of Red Hat OpenShift Container Platform or ETCD is not completed as part of the IBM Cloud Pak for AIOps backup and restore feature.
- Configure the duration between your backups to be less than your RPO. By default, the RPO is 12 hours, which requires backups to run once every 12 hours. The default RTO is 4 hours.
- Custom configuration settings for the Infrastructure Automation - Managed services component, such as Ansible replica count, extra variables, might not be backed up and restored. If you need this data and resources to be included in restored clusters, you need to directly add the data or resources to the restored cluster.
- If you upgrade your deployment, you must setup back and restore again. For more information, see Upgrading Infrastructure Automation backup and restore artifacts.

Backup process

The IBM Cloud Pak® for AIOps backup process is capable of backing up data for both Infrastructure Automation and IBM Cloud Pak for AIOps. The backup of data is completed by backing up the datastores that include Infrastructure Automation data, and if available, IBM Cloud Pak for AIOps data. Some datastores are backed up and restored in their entirety. Other datastores are backed up and restored depending on a particular component need.

The following table shows the IBM Cloud Pak® for AIOps datastores that are backed up:

Database	Backup type
`CouchDB`	Volume backup
`Cassandra`	Data export
`Postgres`	Data export
EDB `Postgres`	Data export
`Minio`	Volume backup
`Elastic`	Partial (Elastic native volume backup used)
`Kafka`	None
`Flink`	None
`mongodb`	Data export
`metastore`	Data export

The backup process is run by `custom backup-job(s)`. When the backup job runs, pods can be scaled down to ensure consistent backups are created. The backup job calls Velero to back up Kubernetes resources and volumes. Native backup scripts are run to back up datastores. The backup job then scales up pods to match their state before the backup process began. Datastores that have a selective backup types are backed up only when the datastore is updated since the previous backup.

For more information about the backup process, see:

- Backing up Infrastructure Automation data.
- Backing up IBM Cloud Pak for AIOps data.

Restore process

To run the restore process, you first need to create a new cluster and install the required prerequisites on the cluster. The prerequisites include installing the required CLI tools, creating the storage classes, and updating the `bcdr/common/aiops-config.json` and `bcdr/restore/restore-data.json` files with the required configuration values. Then you need to configure Velero to point to the same s3 storage where the backup is located.

You then need to create the required namespaces for IBM Cloud Pak® for AIOps by running the `restore-namespace.sh` job. This script creates only the namespaces and namespace metadata, not the contents of the namespace. This namespace restore is needed because the metadata contains SELinux settings that must match the settings from the corresponding namespaces on the cluster where you took the backup.

Then, you need to complete the installation of Infrastructure Automation. With the Infrastructure Automation instance created and Velero configured, you can run the Velero restore jobs to populate the instance. When the jobs complete you need to run any other post restore tasks that you need to complete.

Note: Velero does not support the overriding of resources during the restore. The restoration of resources alread existing in the cluster are skipped. If this occurs, sync the existing resources and create only the resources that were not restored instead of creating a new resources Operator.

For more information about the restore process, see Restoring Infrastructure Automation.

Procedures for backing up and restoring Infrastructure Automation

- Installing the backup and restore tools
- Backing up Infrastructure Automation
- Restoring Infrastructure Automation
- Upgrading Infrastructure Automation backup and restore artifacts
- Backup and Restore Infrastructure management appliances

Installing the backup and restore tools

Learn how to install the tools that are required for the backup and restore of IBM Cloud Pak® for AIOps Infrastructure Automation.

You need to install the Red Hat OpenShift APIs for Data Protection (OADP), Velero, and Restic. Velero and Restic are installed by using the OADP operator. OADP provides default Velero plug-ins that are integrated with storage providers to support backup operations.

Note: The backup and restore tools for Infrastructure Automation are the same as the tools for backing up and restoring IBM Cloud Pak for AIOps. If you have the tools installed for IBM Cloud Pak for AIOps, you do not need to complete the following procedure.

Prerequisites

Ensure that the following tasks are completed before you begin to install the backup and restore tools:

- Provision the object storage (S3 bucket) for storing your backups. The backup process moves backup files to an S3 bucket, which must be in an S3 compatible location, such as Amazon Web Services (AWS).
- Ensure that the following command line tools are installed on your workstation:
  - `kubectl`
  - `oc`
  - `jq`
  - `git`

Procedure

1. Install the Velero client CLI
2. Install OpenShift API for Data Protection (OADP) and set environment variables
3. Download the latest ubi9 image
4. Build and push backup and restore image

1. Install the Velero client CLI

You need to install the Velero client CLI on the workstation that you plan to use to run the scripts for backing up and restoring the cluster.

1. Run the following command to download the Velero CLI client:

   ```
   wget https://github.com/vmware-tanzu/velero/releases/download/v1.9.1/velero-v1.9.1-linux-amd64.tar.gz
   ```

2. Extract the tarball:

   ```
   tar -xvf velero-v1.9.1-linux-amd64.tar.gz -C /tmp
   ```

3. Move the extracted velero binary to `/usr/local/bin`:

   ```
   sudo mv /tmp/velero-v1.9.1-linux-amd64/velero /usr/local/bin
   ```

4. Verify the installation:

   ```
   velero version
   ```

   ```
   Client:
       Version: v1.9.1
       Git commit: e4c84b7b3d603ba646364d5571c69a6443719bf2
   ```

2. Install OpenShift API for Data Protection (OADP) and set environment variables

You need to install Velero and Restic using the Red Hat OpenShift APIs for Data Protection (OADP) Operator.

1. Clone the IBM Cloud Pak for AIOps samples GitHub repository that includes the backup and restore scripts. Clone this repository by running the following command:

   ```
   git clone https://github.com/IBM/cp4waiops-samples.git
   ```

2. Update the install-velero-config.json file.

   1. Go to the directory **<Path>/bcdr/4.2.1/install** where you have the backup and recovery files downloaded. **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.

   2. Open the **install-velero-config.json** file for editing. Configure the following settings in the file:

      - **aws_access_key_id** - Access key ID to connect to your object storage S3 bucket.
      - **aws_secret_access_key** - Secret access key to connect to your S3 bucket.
      - **bucket_name** - Name of the S3 bucket where backup data will be stored.
      - **bucket_region** - Region where the S3 bucket is deployed.
      - **namespace** - Namespace name in which you want to deploy Velero. If the namespace does not exist, it is created when you run the Velero installation script.
      - **backup_label** - Label for creating a folder structure to organize sets of backups together.

3. Log in to the Red Hat OpenShift Container Platform cluster by running the following command:

   ```
   oc login --token=<TOKEN> --server=<URL>
   ```

   Where:

   - **<TOKEN>** is the token that you use to log in to the Red Hat OpenShift Container Platform cluster.
   - **<URL>** is the Red Hat OpenShift Container Platform server URL.

4. Change to the **install** directory where the install script is located:

   ```
   cd <Path>/bcdr/4.2.1/install
   ```

   Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak for AIOps backup and restore files.

5. Run the Velero installation script:

   ```
   nohup ./install-velero.sh > install-velero.log &
   ```

6. Check the logs by running the following command:

   ```
   tail -f install-velero.log
   ```

7. Check the Velero pods status by running the following command:

   ```
   oc get pods -n <veleroNamespace>
   ```

   Where **<veleroNamespace>** is the namespace where Velero is installed.

   The Velero and Restic pods should be in a running state.

8. Check the status of the **backupStorageLocation** by running the following command:

   ```
   oc get backupStorageLocation -n <veleroNamespace>
   ```

   Where **<veleroNamespace>** is the namespace where Velero is installed.

   The **backupStorageLocation** needs to be in an available status.

9. Set environment variables.

   ```
   export IMAGE_REGISTRY_URL=<Image Registry Server URL>
   export REPOSITORY=<Repository>
   export USERNAME=<USERNAME>
   ```

   Where:

   - **<Image Registry Server URL>** is the image registry server URL.
   - **<Repository>** is the repository where you put the image.
   - **<USERNAME>** is the username for logging in to the image registry server.

   Note: You can use any image registry server to store the backup image, such as quay.io, Docker Registry, or any other private image registry. You must have image pull and push access to the image registry server. The image repository must also be accessbile to the cluster where

you plan to run the backup and restore processes.

3. Download the latest ubi9 image

1. Download the latest **ubi9** image from Docker.
2. Run the following command to login to your registry.

   ```
   docker login ${IMAGE_REGISTRY_URL} -u ${USERNAME}
   ```

3. Tag the **ubi19** image.

   ```
   docker tag redhat/ubi9:latest ${IMAGE_REGISTRY_URL}/${REPOSITORY}/ubi9:latest
   ```

4. Push the **ubi9** image to your repository.

   ```
   docker push ${IMAGE_REGISTRY_URL}/${REPOSITORY}/ubi9:latest
   ```

5. Update the YAML files that you cloned to reference the new **ubi9** image.

   Update **spec.image** to be **<registry>/ubi9:latest** in the following YAML files, where **<registry>** is **${IMAGE_REGISTRY_URL}/${REPOSITORY}**

   - backup/scripts/cam/cam-backup.yaml
   - backup/scripts/common-services/dummy-db.yaml
   - backup/scripts/elasticsearch/es-backup-pod.yaml
   - backup/scripts/metastore/metastore-backup.yaml
   - backup/scripts/other-resources/other-resources-backup.yaml
   - backup/scripts/postgres/postgres-backup.yaml
   - backup/scripts/sts-backup-pod-template.yaml
   - restore/common-services/cleanup-pod.yaml

4. Build and push backup and restore image

1. Build the Docker image.
   1. Change to the directory where you have the backup and recovery files downloaded.

      ```
      cd <Path>/bcdr/4.2.1/
      ```

      Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak for AIOps backup and restore files.

   2. Build the **cp4waiops-bcdr** Docker image by running following command:

      ```
      docker build -t cp4waiops-bcdr:v4.2.1 -f Dockerfile .
      ```

2. Tag and push the Docker image to the image registry.
   1. Run the following command:

      ```
      docker tag cp4waiops-bcdr:v4.2.1 ${IMAGE_REGISTRY_URL}/${REPOSITORY}/cp4waiops-bcdr:v4.2.1
      ```

   2. Run the following command:

      ```
      docker push ${IMAGE_REGISTRY_URL}/${REPOSITORY}/cp4waiops-bcdr:v4.2.1
      ```

Troubleshooting

- Velero installation failed
- Velero backups are not showing after installation

Velero installation failed

If your installation of Velero fails for any reason, delete any currently installed Velero files and reattempt the installation.

For example, the install script **install-velero.sh** might not complete successfully. If this occurs an error similar to the following message can display:

```
error: unable to recognize "STDIN": no matches for kind "DataProtectionApplication" in version
"oadp.openshift.io/v1alpha1"
```

This can occur if the network or cluster is running slow and taking too long to pull the required images. This delay causes the OADP operator to take too long to create the required custom resource.

If the preceding issue occurs, or the install fails for another reason, run the following **oc** commands to delete any existing Velero installation files before attempting to install again. These commands use **<veleroNamespace>** as the project (namespace) where Velero is installed. Replace this value with the project where you are attempting to install Velero.

1. Delete any cluster service versions (CSV) in the project:

   ```
   oc delete csv --all -n <veleroNamespace>
   ```

2. Delete any subscriptions in the project:

   ```
   oc delete subscription --all -n <veleroNamespace>
   ```

3. Delete any deployments in the project:

   ```
   oc delete deploy --all -n <veleroNamespace>
   ```

4. Delete any daemonsets in the project:

   ```
   oc delete ds --all -n <veleroNamespace>
   ```

5. Delete the project (namespace):

   ```
   oc delete project <veleroNamespace>
   ```

Any existing installation is deleted and you can reattempt the installation.

Velero backups are not showing after installation

If your Velero backups do not show, complete the following steps:

1. Correct the **bucket** and **prefix** property within the **backupstoragelocation** resource that is named **bcdr-s3-location**. The **bucket** and **prefix** name should match the value that is defined within the **install-velero-config.json** file during the Velero installation.

   Run the **oc edit** command to edit the **backupstoragelocation** resource:

   ```
   oc edit backupstoragelocation bcdr-s3-location -n <veleroNamespace>
   ```

   Where **<veleroNamespace>** is the namespace where Velero is installed.

2. Delete the Velero pod by running the following command:

   ```
   oc delete pod <pod> -n <veleroNamespace>
   ```

   Where **<veleroNamespace>** is the namespace where Velero is installed, and **<pod>** is the name of the Velero pod.

   The backups should now be available.

3. Run the following command to list the backups:

   ```
   velero get backup
   ```

Backing up Infrastructure Automation data

Learn how to back up data for Infrastructure Automation components.

The backup process uses Velero for cluster object backups and uses Restic for volume backups. The backup script must be the same version as your deployment.

- Prerequisites
- Procedure
- Troubleshooting

Note: If you also need to back up data for IBM Cloud Pak for AIOps components, the process and tools to use are the same. The only change that is required is when you are packaging the Helm chart. During this step you need to configure the value of the **enabledComponents** parameter to include **'"AIOPS"'** as well as **'"IA"'**. All other steps remain the same.

Prerequisites

- Provision S3 compliant object storage for your backups. You must have sufficient storage to backup the data that is contained in the datastores and persistent volume claims (PVCs) of the components that are backed up, as detailed in Backup process. The storage must also be accessible by any environment that a backup might need to be restored to. The backup process moves backup files to an S3 bucket,

which must be in an S3 compatible location. For instance, you can use the following public or private cloud options for provisioning storage for your backups:

  - MinIO
  - Red Hat OpenShift Data Foundation object store
  - IBM Cloud Object Storage
  - AWS Cloud

Note: As your data grows, the size of your backup storage might need to grow, and you must ensure that sufficient storage for backups is provisioned to accommodate any growth in data size.

- If they are not already installed, install the backup and restore tools. You need to install the Red Hat OpenShift APIs for Data Protection (OADP), Velero, and Restic in the Red Hat OpenShift Container Platform cluster. For more information, see Installing the backup and restore tools.

  Important: Ensure that the OpenShift APIs for Data Protection (OADP) and Velero are configured to point to the same object storage (S3 bucket) that includes the backup that you plan to use.

Procedure

1. Clone the IBM Cloud Pak for AIOps samples GitHub repository to retrieve the latest backup and restore scripts. Clone this repository by running the following command:

   ```
   git clone https://github.com/IBM/cp4waiops-samples.git
   ```

2. Log in to the Red Hat OpenShift Container Platform cluster by running the following command:

   ```
   oc login --token=<TOKEN> --server=<URL>
   ```

   Where:

   - **<TOKEN>** is the token that you use to log in to the Red Hat OpenShift Container Platform cluster.
   - **<URL>** is the Red Hat OpenShift Container Platform server URL.

3. Create an image pull secret by running the following command:

   ```
   oc create secret docker-registry backup-secret -n velero --docker-server=<Image Registry Server URL> --docker-username=<USERNAME> --docker-password=<PASSWORD> --docker-email=<EMAIL>
   ```

   Where:

   - **<Image Registry Server URL>** is the image registry server URL.
   - **<USERNAME>** is the username to log in to the image registry server.
   - **<PASSWORD>** is the password to log in to the image registry server.
   - **<EMAIL>** is the email for image registry server.

4. Package the Helm Chart

   1. Change to the directory where you have the backup and recovery files downloaded.

      ```
      cd <Path>/bcdr/4.2.1/backup
      ```

      Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.

   2. Update the following parameters in the **values.yaml** file. The file is located in the **./helm** directory.

      - **repository** - The name of the Docker image.

        For example, **xy.abc.io/cp4waiops/cp4waiops-bcdr**. In this example, **xy.abc.io** is the image registry server URL, **cp4waiops** is the name of the repository and **cp4waiops-bcdr** is the name of the Docker image.

      - **pullPolicy** - The policy for determining when to pull the Docker image from the image registry server.

        For example, to force pull the image, use the **Always** policy.

      - **tag** - The tag for the Docker image. For example, **latest**.

      - **pullSecret** - The name of the image pull secret.

      - **schedule** - The Cron expression for the automated backup. For example, to take a backup once a day, use the **0 0 * * *** Cron expression.

      - **backupStorageLocation** - The **velero** storage location where backed up data is stored. For example, **bcdr-s3-location**.

        Use the **oc get backupstoragelocation -n <velero-namespace>** command to get the **backupStorageLocation** on the OpenShift cluster.

- **backupNameSuffix** - The prefix for the backup name when the backup is created by using the job. Generally, it can be name of source cluster itself. For example, **aiops-cluster-backup-106**.
- **aiopsNamespace** - The namespace where IBM Cloud Pak for AIOps is installed.
- **csNamespace** - The namespace where IBM Cloud Pak foundational services is installed.
- **veleroNamespace** - The namespace where Velero is installed.
- **ttl** - The time-to-live setting for the backup. The backed up data is retained until the **ttl** value is reached (expires). For example **720h0m0s**.
- **enabledNamespaces** The namespaces where components that need to be backed up are installed.

  For example, the **ibm-common-services** namespace represents the IBM Cloud Pak foundational services component. You can delete any unused namespaces from the list to reduce the time that is required for the backup process. You can update the list as shown if you have installed only two components, for example, IBM Cloud Pak foundational services and IBM Cloud Pak for AIOps

  ```
  enabledNamespaces:
  - '"ibm-common-services"'
  - '"cp4aiops"'
  ```

  Component: IBM Cloud Pak foundational services

  - Namespaces: **ibm-common-services**

  Component: IBM Cloud Pak for AIOps

  - Namespaces: **cp4aiops**
- **enabledComponents** - Required. The list of components to back up.

  The backup and restore processes support backing up and restoring both the IBM Cloud Pak for AIOps and Infrastructure Automation components. These components can be installed together or independtly. You can specify to back up both, or only one of these components. Any other value is ignored and a corresponding error message is generated.

  To specify Infrastructure Automation, set the value **'"IA"'**.

  To specify IBM Cloud Pak for AIOps, set the value **'"AIOPS"'**.

  The following configuration shows both components listed:

  ```
  enabledComponents:
  - '"IA"'
  - '"AIOPS"'
  ```

3. Package the Helm Chart.

   ```
   helm package ./helm
   ```

5. Trigger an automated backup

   1. Change to the **backup** directory where the script is located:

      ```
      cd <Path>/bcdr/4.2.1/backup
      ```

      Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak for AIOps backup and restore files.

   2. Deploy the backup job by running the following command:

      ```
      helm install backup-job clusterbackup-0.1.0.tgz
      ```

6. Monitor the backup Job

   1. Check the backup pods status by running the following command:

      ```
      oc get pods -n velero
      ```

   2. Check the backup job logs by running the following command:

      ```
      oc logs -f <backup-job-***>
      ```

   3. Check the backup status by running the following command:

      ```
      velero get backup <BACKUP_NAME>
      ```

      Where:

- **<BACKUP_NAME>** is the name of the Backup. You can see the backup name after the backup job is complete. For example, you might see the backup name **aiops-cluster-backup-106-1622193915** in the backup job log as follows:

  ```
  Waiting for backup aiops-cluster-backup-106-1622193915 to complete
  ```

4. Review the backup result summary that is included within the logs of the backup job.

   Note: A **backup-result-config** ConfigMap is created to include the result summary. To view this ConfigMap, run the following command:

   ```
   oc get cm backup-result-config -n <namespace> -o yaml
   ```

   Where **<namespace>** is the project (namespace) where Infrastructure Automation is installed.

On-demand backups

This step is optional. Use only when you do not want to wait untill the next scheduled backup job.

Prerequisite: The deployment of an automated backup job is a prerequisite for the on-demand job. Only after you initiate an automated backup job, then you can trigger an on-demand backup.

1. Deploy the on-demand backup job by running the following command:

   ```
   oc create job --from=cronjob/backup-job on-demand-backup-job -n velero
   ```

2. Check the on-demand backup pods status by running the following command:

   ```
   oc get pods -n velero
   ```

3. Check the on-demand backup job logs by running the following command:

   ```
   oc logs -f <on-demand-backup-job-***>
   ```

4. Check the backup status by running the following command:

   ```
   velero get backup <BACKUP_NAME>
   ```

   Where <BACKUP_NAME> is the name of the Backup. You can see the backup name after the on-demand backup job is complete. For example, you might see the backup name **aiops-cluster-backup-106-1622193915** in the on-demand backup job log as follows:

   ```
   Waiting for backup aiops-cluster-backup-106-1622193915 to complete
   ```

   The status is in **completed** state when the backup is finished, such as in the following sample output:

   ```
   aiops-backup-1654171690-testfest34   Completed                   2022-06-02 05:08:11 -0700 PDT   29d
   bcdr-data-protection-app-1   cp4aiops.ibm.com/backup=t
   ```

5. Review the backup result summary that is included within the logs of the backup job.

   Note: A **backup-result-config** ConfigMap is created to include this result. To view this ConfigMap, run the following command:

   ```
   oc get cm backup-result-config -n <namespace> -o yaml
   ```

   Where **<namespace>** is the project (namespace) where Infrastructure Automation is installed.

Troubleshooting

- Velero backups are stuck in the *In progress* state
- Classifier and layout pods remain not ready after running a backup
- Helm install backup job command failed

Velero backups are stuck in the *In progress* state

If your Velero backups remain stuck in the *In progress* state for an unexpected duration, complete the following steps. This procedure stops the backup process so that you can try the backup again.

1. Delete the Velero pod by running the following command:

   ```
   oc delete pod <pod> -n <veleroNamespace>
   ```

   Where **<veleroNamespace>** is the namespace where Velero is installed, and **<pod>** is the name of the Velero pod.

2. Run the backup that is stuck in the *In progress* state:

   ```
   velero delete backup <backup>
   ```

Where **<backup>** is the backup that you want to delete. The backup process should begin again.

3. Wait for the backup to complete and verify that the backup is created.

Classifier and layout pods remain not ready after running a backup

After you complete a backup, you might notice that the classifier (**aiops-ir-analytics-classifier**) and layout (**aiops-topology-layout**) pods do not change to a ready state, which can cause **AIOpsAnalyticsOrchestrator** to be stuck in an updating phase. This issue can occur after most backups. To resolve this issue, you need to either restart the pods or increase the resource limits for the pods.

- For the classifier (**aiops-ir-analytics-classifier**) pod, restart the pod.
- For the layout (**aiops-topology-layout**) pod, restart the pod. If the restart does not result in the pod becoming ready, increase the resource limits for the pod. Doubling the **cpu** and **memory** limits and request allocations for the pods can result in the pod starting and entering a ready state.

Helm install backup job command failed

When you are running the **helm install backup-job clusterbackup-0.1.0.tgz** command, you might encounter the command failing with an error that is similar to the following error:

```
Error: admission webhook "trust.hooks.securityenforcement.admission.cloud.ibm.com" denied the request:
Deny "<Image Registry Server URL>/<Repository>/cp4waiops-bcdr:latest", no matching repositories in
ClusterImagePolicy and no ImagePolicies in the "velero" namespace
```

Where:

- **<Image Registry Server URL>** is the image registry server URL.
- **<Repository>** is the repository where you put the image.

If you encounter this error, complete the following steps to resolve the issue:

1. Uninstall the **backup-job** job by running the following command:

   ```
   helm uninstall backup-job -n velero
   ```

2. Create a **backup-image-policy.yaml** file and add the following content within the file:

   ```
   apiVersion: securityenforcement.admission.cloud.ibm.com/v1beta1
   kind: ClusterImagePolicy
   metadata:
     name: backup-image-policy
   spec:
    repositories:
     - name: "<Image Registry Server URL>/<Repository>/cp4mcm-bcdr:latest"
       policy:
   ```

3. Apply the policy by running the following command:

   ```
   oc apply -f backup-image-policy.yaml
   ```

4. Deploy the backup job by running the following command:

   ```
   helm install backup-job clusterbackup-0.1.0.tgz
   ```

Restoring Infrastructure Automation data

Learn how to restore data for Infrastructure Automation components to a cluster, such as for disaster recovery.

The following procedure restores all backed up data that exists in the specified backup for Infrastructure Automation components. The steps in the following procedure restore Infrastructure Automation in a new cluster.

Notes:

- All required storage classes must be created before you run the restore process. The storage classes must have the same name as the backup cluster.
- Custom configuration settings for the Infrastructure Automation - Managed services component, such as Ansible replica count, extra variables, might not be backed up and restored. If you need this data and resources to be included in restored clusters, you need to directly add the data or resources to the restored cluster.
- You can restore a backup only within an environment that has the same version of Infrastructure Automation as the environment where the backup was created. For example, a backup of an Infrastructure Automation 4.2.1 environment must be restored within a cluster that has Infrastructure Automation 4.2.1 installed. If you need to upgrade and restore data, complete the restore process before you upgrade.

- If you are also restoring IBM Cloud Pak for AIOps data, the overall procedure is mostly the same, with the following additional steps required:
  - You need to first install IBM Cloud Pak for AIOps on the cluster where you are restoring data. Then, you need to install Infrastructure Automation.
  - You need to first restore your IBM Cloud Pak for AIOps data. Then, you can restore your Infrastructure Automation data.

  For more information, see Restoring IBM Cloud Pak for AIOps.

Procedure

The process for restoring Infrastructure Automation to a cluster involves the following step sections:

1. Set up your new cluster for backup and restore.

   Your first steps for restoring a cluster requires you to install OpenShift, configure storage, and install the backup and restore tools.

2. Prepare the backup data for restoring.

   Before you can run any restore jobs, you need to verify your backed-up data and prepare the data for restoring.

3. Restore the cluster namespaces and install Infrastructure Automation

   Since the restore job does not install Infrastructure Automation, you need to first install Infrastructure Automation before you can run the restore jobs for restoring database and component data. A restore job for restoring the cluster namespaces is available and must be run before you install Infrastructure Automation.

4. Optional. Restore IBM Cloud Pak for AIOps data

5. Restore the Infrastructure Automation data

   This procedure restores the data for all backed up Infrastructure Automation databases and components. Optionally, you can choose to restore only individual components. For more information, see Restoring individual components.

If you encounter any issues with the restore process, see Troubleshooting

1. Set up your new cluster for backup and restore

1. Install OpenShift and create a cluster by using the instructions in Installing Red Hat OpenShift Container Platform.

   IBM Cloud Pak for AIOps requires OpenShift to be installed and running. You must have administrative access to your OpenShift cluster.

   Important: Ensure that the version of Red Hat OpenShift Container Platform that you install is the same as the version that was installed in the backed-up cluster.

   For information about the supported versions of OpenShift, see Supported Red Hat OpenShift Container Platform versions. The hardware architecture that you install IBM Cloud Pak for AIOps on must be AMD64.

   Note: Infrastructure Automation uses the OpenShift image registry when it builds images in real time. If the OpenShift image registry is not persistent and the registry respawns, then workloads can temporarily fail until respawning is complete. A persistent OpenShift image registry is recommended to avoid this issue. For more information, see Setting up and configuring the registry in the Red Hat OpenShift Container Platform documentation.

2. Install the OpenShift command-line interface (`oc`) on your cluster's boot node and run `oc login`, using the instructions in Getting started with the Red Hat OpenShift CLI.

3. Configure storage

   The storage configuration must satisfy your sizing requirements. For more information about the storage classes that are needed for installing IBM Cloud Pak for AIOps, see Storage.

   Important: All required storage classes must be created before running the restore process. The storage classes must have the same name as the backup cluster.

4. Install the backup and restore tools

   1. Install the Red Hat OpenShift APIs for Data Protection (OADP), Velero, and Restic in the Red Hat OpenShift Container Platform cluster. For more information, see Installing the backup and restore tools.

      Important: Ensure that the Red Hat OpenShift APIs for Data Protection (OADP) and Velero are configured to point to the same object storage (S3 bucket) that includes the backup that you plan to use.

2. Prepare the backup data for restoring

1. Check the backup status

   Check the backup status to ensure that the backup that you want to restore in your cluster is complete. Run the following command to check the contents of the backup:

   ```
   velero describe backup <backup-name> --details
   ```

   The output should list the backed-up data for Infrastructure Automation and for IBM Cloud Pak foundational services (**ibm-common-services/***). If you also backed up IBM Cloud Pak for AIOps data, this data (**cp4waiops/***) should also be listed.

2. Create an image pull secret by running the following command to create the image pull secret:

   ```
   oc create secret docker-registry restore-secret -n velero --docker-server=<Image Registry Server URL> --docker-username=<USERNAME> --docker-password=<PASSWORD> --docker-email=<EMAIL>
   ```

   Where:

   - **<Image Registry Server URL>** is the image registry server URL.
   - **<USERNAME>** is the username to log in to the image registry server.
   - **<PASSWORD>** is the password to log in to the image registry server.
   - **<EMAIL>** is the email for image registry server.

3. Package and install the Helm Chart

   1. Change to the **restore** directory where you need to package the Helm Chart:

      ```
      cd <Path>/bcdr/4.2.1/restore
      ```

      Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.

   2. Update the following parameters in the **values.yaml** file. The file is located in the **./helm** directory:

      - **backupName** - The name of the backup that you are restoring.
      - **aiopsNamespace** - The namespace where IBM Cloud Pak for AIOps is installed.
      - **csNamespace** - The namespace where IBM Cloud Pak foundational services is installed.
      - **veleroNamespace** - The namespace where Velero is installed.

   3. Package the Helm Chart.

      ```
      helm package ./helm
      ```

   4. Install the Helm Chart for restoring data by running the following job:

      ```
      helm install restore-job clusterrestore-0.1.0.tgz
      ```

3. Restore the cluster namespaces and install Infrastructure Automation

1. Update the image and job names

   1. Change to the **restore** directory where the restore script is located:

      ```
      cd <Path>/bcdr/4.2.1/restore
      ```

      Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.

   2. Open the following YAML files for editing:

      - **ns-restore-job.yaml** - This file is used for the job to restore the cluster namespaces
      - **ia-restore-job.yaml** - This file is used for the job to restore Infrastructure Automation.
      - Optional. **aiops-restore-job.yaml** - This file is used for the job to restore IBM Cloud Pak for AIOps. If you do not need to restore IBM Cloud Pak for AIOps data, you can skip updating this file.

   3. Update the image name in each YAML file. You must specify the name of the image that you have pushed to the image repository: **<Image Registry Server URL>/<Repository>/cp4waiops-bcdr:latest**.

   4. Update the **JOB_NAME** value in each YAML file:

      - For the **ns-restore-job.yaml** file, update the value to be **ns-restore-job**.
      - For the **ia-restore-job.yaml** file, update the value to be **ia-restore-job**.
      - Optional. For the **aiops-restore-job.yaml** file, update the value to be **aiops-restore-job**.

   5. Save your changes.

2. Restore the cluster namespaces

   You need to restore the projects (namespaces) of the backed-up cluster so that your new cluster includes the metadata with the SELinux settings that need to match the settings for the backup data that you plan to restore.

   These steps only restore the namespaces and namespace metadata. The commands do not restore the contents of the namespaces.

1. Change to the **restore** directory where the restore script is located:

   ```
   cd <Path>/bcdr/4.2.1/restore
   ```

   Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.

2. Optional. Delete any existing namespace restore jobs:

   ```
   oc delete -f ns-restore-job.yaml
   ```

3. Create a job to restore the cluster namespaces:

   ```
   oc create -f ns-restore-job.yaml
   ```

4. Check the restore job logs by running the following command:

   ```
   oc logs -f <ns-restore-job-***>
   ```

5. Check the velero-restore status for the namespace by running the following command:

   ```
   velero get restore <RESTORE_NAME>
   ```

   Where **<RESTORE_NAME>** is the name of the restore for namespace. You can view the restore name after the restore job is completed. For example, you might see the restore name **cs-namespace-restore-20221006054710** and **aiops-namespace-restore-20221006054710** within the restore job log as follows:

   ```
   Restore request "cs-namespace-restore-20221006054710" submitted successfully.
   ```

   and

   ```
   Restore request "aiops-namespace-restore-20221006054710" submitted successfully.
   ```

Ensure that the projects (namespaces) are restored before you proceed.

3. Optional. Install IBM Cloud Pak for AIOps

   If you also need to restore IBM Cloud Pak for AIOps data, you need to install IBM Cloud Pak for AIOps before you install Infrastructure Automation. For more information, see Installing IBM Cloud Pak for AIOps.

   - For more information about restoring IBM Cloud Pak for AIOps, see Restoring IBM Cloud Pak for AIOps.

   Important: Ensure that the version of IBM Cloud Pak for AIOps that you are installing is the same as the version that was installed in the backed-up cluster.

4. Install Infrastructure Automation.

   You need to install the Infrastructure Automation operators and create the required custom resources.

   - For more information about installing Infrastructure Automation, see Installing Infrastructure Automation.

   Note: When you install Infrastructure Automation, the Infrastructure Automation operator and the IAConfig CR are created.

   - If you install the Managed services component, the Managed Services operator is also installed and the corresponding custom resource (CR) is automatically created.
   - If you install the Infrastructure management component, only the Infrastructure management operator is installed. The corresponding CR is not created. You do not need to create this CR as it is created during the restore process in the following steps.

Important:

- The backup includes keys and certificates from the backed-up cluster. Ensure that your new cluster is configured to support the use of these keys and certificates so that the restored data can be accessed.
- Wait until the installation is complete and all pods in the IBM Cloud Pak for AIOps project (namespace) are running before you proceed with the following restore steps.

4. Optional. Restore the IBM Cloud Pak for AIOps data

If you are also restoring IBM Cloud Pak for AIOps data, run the commands to restore the IBM Cloud Pak for AIOps data before you restore your Infrastructure Automation data. For more information, see Restoring IBM Cloud Pak for AIOps.

5. Restore the Infrastructure Automation data

This procedure restores the data for all backed up Infrastructure Automation databases and components. Optionally, you can choose to restore only individual components. For details, see Restoring individual components.

1. Change to the **restore** directory where the restore script is located:

   ```
   cd <Path>/bcdr/4.2.1/restore
   ```

   Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.

2. Optional. Delete any existing Infrastructure Automation restore jobs:

   ```
   oc delete -f ia-restore-job.yaml
   ```

3. Create a job to restore Infrastructure Automation:

   ```
   oc create -f ia-restore-job.yaml
   ```

4. Check the restore job logs by running the following command:

   ```
   oc logs -f <ia-restore-job-***>
   ```

5. Check the velero-restore status by running the following command:

   ```
   velero get restore <RESTORE_NAME>
   ```

   Where **<RESTORE_NAME>** is the name of the restore for Infrastructure Automation. You can view the restore name after the restore job is completed. For example, you might see the restore name **cam-restore-20221006054710** for the Managed services restore in the restore job log as follows:

   ```
   Restore request "cam-restore-20221006054710" submitted successfully.
   ```

   Similarly, the velero restore name for other components can display in the restore job log.

6. When the restore is completed, and all **infrastructure-management** pods are in a running started, restart the **zen-watcher** pod. Run the following command to restart the pod:

   ```
   oc delete pod -l app.kubernetes.io/component=zen-watcher -n <aiopsNamespace>
   ```

   Where **aiopsNamespace** is the namespace where IBM Cloud Pak for AIOps is installed.

Restoring individual components

If needed, you can choose to restore data for specific individual databases and components instead of for all databases and components at the same time.

1. Change to the **restore** directory where the restore script is located:

   ```
   cd <Path>/bcdr/4.2.1/restore
   ```

   Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.

2. Copy the **ia-restore-job.yaml** file. Rename your new file **<component>-restore-job.yaml**, where **<component>** is the name of the component that you are restoring.

   For example, if you are restoring Managed services, rename the file to be **cam-restore-job.yaml**. If you are restoring Infrastructure management, rename the file to be **im-restore-job.yaml**.

3. Open the new **<component>-restore-job.yaml** file for editing. Update the **name** and **command** sections to match the values for the component that you are restoring:

   - Update the **name** of the restore job in the metadata section to be the individual component job, such as **cam-restore-job**.
   - Update the command section **command: ["/bin/bash", "restore.sh","-ia"]** to replace **-ia** with the respective argument for the component that you are restoring. For the list of component arguments, see the table that follows this procedure.

4. Create a job to restore the individual component.

   ```
   oc create -f <component>-restore-job.yaml
   ```

5. Check the restore job logs by running the following command:

   ```
   oc logs -f <cp4waiops-component-restore-job-***>
   ```

6. Check the velero-restore status by running the following command:

   ```
   velero get restore <RESTORE_NAME>
   ```

   Where **<RESTORE_NAME>** is one of the restore names for the component. You can view the names for the component in the restore job log when the restore job is completed. For example, the restore name for a Managed services restore can be **cam-restore-20221006054710**, which can display in an entry similar to the following example log entry:

   ```
   Restore request "cam-restore-20221006054710" submitted successfully.
   ```

Component or database arguments for restore job command configuration

Table. Component or database arguments for restore job commands

Component or Database	Argument
`Cassandra`	`-cassandra`
`CouchDB`	`-couchdb`
`Elasticsearch`	`-es`
`Metastore`	`-metastore`
`Minio`	`-minio`
`Postgres`	`-postgres`
IBM Cloud Pak foundational services	`-cs`
Managed services	`-cam`
Infrastructure management	`-im`

Troubleshooting

- LDAP user login is not working after a restore
- Velero restore is stuck in the *In progress* state
- Helm install restore job command failed
- Restore process terminated mid-process with partial data available
- Managed services instance deployment fails due to a socket hang up
- Infrastructure management pods are not running after a restore
- Infrastructure management URL in the navigation panel points to the cluster where the backup was taken
- Troubleshooting the IBM Cloud Pak for AIOps restore

LDAP user login is not working after a restore

Follow the steps to solve the problem:

1. Log in to the console as the default admin user.
2. From the main navigation menu, click **Administer** > **Identify and access**.
3. Select the LDAP connection, and click **Edit connection**. Edit the LDAP connection with the correct information.
4. Click **Test connection**.
5. Click **Save** once the connection is success.
6. log in to the console with the LDAP user's credentials.

Velero restore is stuck in the *In progress* state

If your Velero backups remain stuck in the *In progress* state for an unexpected duration, complete the following steps. This procedure stops the backup process so that you can try the backup again.

1. Delete the Velero pod by running the following command:

   ```
   oc delete pod <pod> -n <veleroNamespace>
   ```

   Where **<veleroNamespace>** is the namespace where Velero is installed, and **<pod>** is the name of the Velero pod.

2. Delete the restore that is stuck in the *In progress* state:

   ```
   velero delete restore <restore>
   ```

   Where **<restore>** is the restore process that you want to delete. The process should begin again.

3. Wait for the process to complete and verify that the restored data is available.

Helm install backup job command failed

When you are running the **`helm install restore-job clusterrestore-0.1.0.tgz`** command, you might encounter the command failing with an error that is similar to the following error:

```
Error: admission webhook "trust.hooks.securityenforcement.admission.cloud.ibm.com" denied the request:
Deny "<Image Registry Server URL>/<Repository>/cp4waiops-bcdr:latest", no matching repositories in
ClusterImagePolicy and no ImagePolicies in the "velero" namespace
```

Where:

- **`<Image Registry Server URL>`** is the image registry server URL.
- **`<Repository>`** is the repository where you put the image.

If you encounter this error, complete the following steps to resolve the issue:

1. Uninstall the **`restore-job`** job by running the following command:

   ```
   helm uninstall restore-job -n velero
   ```

2. Create a **`restore-image-policy.yaml`** file and add the following content within the file:

```
apiVersion: securityenforcement.admission.cloud.ibm.com/v1beta1
kind: ClusterImagePolicy
metadata:
  name: restore-image-policy
spec:
 repositories:
  - name: "<Image Registry Server URL>/<Repository>/cp4mcm-bcdr:latest"
    policy:
```

3. Apply the policy by running the following command:

   ```
   oc apply -f restore-image-policy.yaml
   ```

4. Deploy the backup job by running the following command:

   ```
   helm install restore-job clusterbackup-0.1.0.tgz
   ```

Restore process terminated mid-process with partial data available

If the restore process does not complete as expected for a data store, such as if it is aborted or terminated during its run, data might not be restored correctly. This incomplete restored data neeeds to be removed before you run the restore process again. To remove the data, you need to run a post-restore cleanup script to clean up the data store.

To run a script, complete the following steps:

1. Define the following environment variable on your workstation:

   ```
   export WORKDIR="<Path>/bcdr/4.2.1/"
   ```

   Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.

2. Change to the **restore** directory where the post-restore cleanup script is located:

   ```
   cd <Path>/bcdr/4.2.1/restore/<data_store>/
   ```

   Where

   - **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak® for AIOps backup and restore files.
   - **<data_store>** is the directory for the data store or custom resource that needs to be cleaned up. For example, **couchdb** is the directory for CouchDb.

3. Run the post-restore cleanup script:

   ```
   nohup ./<data-store>-post-restore.sh.sh > <data-store>-post-restore.log &
   ```

   Where **<data_store>** is the data store or custom resource that needs to be cleaned up.

4. Run the restore process for that data store or resource again.

For example, if the restore job for restoring the Cassandra data store aborted, you need to run the **cassandra-post-restore.sh** post-restore script that is stored in the **bcdr/restore/cassandra** directory to clean the data. Then, run the **cassandra-native-post-restore.sh** post-restore script.

The following table lists the cleanup script to run for each data store:

Table. Cleanup script for components

Component or Database	Cleanup script
Cassandra	Run first: **bcdr/4.2.1/restore/cassandra/cassandra-post-restore.sh** Then, run: **bcdr/4.2.1/restore/cassandra/cassandra-native-post-restore.sh**
CouchDB	**bcdr/4.2.1/restore/couchdb/couchdb-post-restore.sh**
Elasticsearch	**bcdr/4.2.1/restore/elasticsearch/es-post-restore.sh**
Metastore	**bcdr/4.2.1/restore/metastore/metastore-post-restore.sh**
Minio	**bcdr/4.2.1/restore/minio/minio-post-restore.sh**
Postgres	**bcdr/4.2.1/restore/postgres/postgres-post-restore.sh**
IBM Cloud Pak foundational services	**bcdr/4.2.1/restore/common-services/cs-post-restore.sh**
Managed services	**bcdr/4.2.1/restore/cam/cam-post-restore.sh**
Infrastructure Management	**bcdr/4.2.1/restore/infrastructure-management/im-cleanup-restore.sh**

Managed services instance deployment fails due to a socket hang up

After your restore a Managed services (**cam**) backup and Managed services is deployed, the Managed services instance can fail due to a socket hang up issue. If this issue ocurs, restart the **cam-iaas** pod:

```
oc delete pod <cam-iaas-xxxx> -n <aiopsNamespace>
```

Where **aiopsNamespace** is the namespace where IBM Cloud Pak for AIOps is installed.

Infrastructure management pods are not running after a restore

Following a restore, you can notice that the Infrastructure Management pods (prefixed with "1-") are showing as not running.

For example,

```
oc get pod |grep postgresql
postgresql-6cff46dcdc-g5cn8                                       0/1     Running     0                33m
```

This issue can occur when Postgres is not fully initialised. To resolve this issue, if the pod is not showing a ready state (`1/1`), restart the pod manually. This restart enables the Infrastructure Management pods to start.

Infrastructure management URL in the navigation panel points to the cluster where the backup was taken

Following a restore, the Infrastructure Management URL in the navigation panel points to the cluster where the backup was taken. To resolve this issue, restart the zen watcher pod on the restored cluster:

```
oc delete pod -l app.kubernetes.io/component=zen-watcher
```

Troubleshooting the IBM Cloud Pak for AIOps restore

If you are also restoring IBM Cloud Pak for AIOps data and encounter an issue with the restore process for IBM Cloud Pak for AIOps or encounter an issue with data not being available or processed after the restore, see Troubleshooting IBM Cloud Pak for AIOps restore.

Upgrading IBM Cloud Pak for AIOps backup and restore artifacts

Use this procedure if you want to run backup or restore on an upgraded IBM Cloud Pak® for AIOps deployment.

If you run a backup or restore on an IBM Cloud Pak for AIOps deployment, the backup and restore scripts must always be the same version as your deployment. For example, if you run backup on a IBM Cloud Pak for AIOps v4.2.1 deployment, you must use a v4.2.1 backup script.

If you previously set up backup or restore on a deployment that you have now upgraded, then you must clean up any existing backup or restore artifacts. If you then want to run a new backup or restore, you must get the new version of the backup or restore scripts.

Note: If you have never set up backup or restore and you want to run backup or restore, then use the instructions in IBM Cloud Pak for AIOps backup and restore.

Before you begin

Create environment variables for use in the commands in the following steps.

```
export IMAGE_REGISTRY=<image_registry>
export USERNAME=<username>
export PASSWORD=<passowrd>
export EMAIL=<email>
export REPOSITORY=<repository>
export VELERO_NAMESPACE=<velero_ns>
```

Where:

- `<image_registry>` is the image registry server URL.
- `<username>` is the username to log in to the image registry server.
- `<password>` is the password to log in to the image registry server.
- `<email>` is the email for image registry server.
- `<repository>` is the repository where you want to put the images.
- `<velero_ns>` is the namespace that Velero is installed in.

Set up backup on an upgraded IBM Cloud Pak for AIOps deployment

Use the following steps to upgrade your backup artifacts.

1. Uninstall the backup helm chart.

   Run the following command in the namespace that **Red Hat OpenShift APIs for Data Protection** (OADP) is installed in.

   ```
   helm uninstall backup-job
   ```

2. If your image registry server is unchanged since you previously set up backup, then skip this step. Otherwise, delete and create a new image pull secret for backup.

```
oc delete secret backup-secret -n ${VELERO_NAMESPACE}
oc create secret docker-registry backup-secret -n ${VELERO_NAMESPACE} --docker-server=${IMAGE_REGISTRY}
--docker-username=${USERNAME} --docker-password=${PASSWORD} --docker-email=${EMAIL}
```

3. Clone the IBM Cloud Pak for AIOps samples GitHub repository that holds the backup and restore scripts. Clone this repository by running the following command:

   ```
   git clone https://github.com/IBM/cp4waiops-samples.git
   ```

4. Build the Docker image.

   1. Change to the directory that you downloaded the backup and recovery files to in the previous step.

      ```
      cd <Path>/bcdr/4.2.1/
      ```

      Where **<Path>** is the path to where you downloaded and extracted the IBM Cloud Pak for AIOps backup and restore files.

   2. Build the **cp4waiops-bcdr** Docker image by running following command:

      ```
      docker build -t cp4waiops-bcdr:v4.2.1 -f Dockerfile .
      ```

5. Tag and push the Docker image to the image registry.

   1. Run the following command to tag your image:

      ```
      docker tag cp4waiops-bcdr:v4.2.1 ${IMAGE_REGISTRY}/${REPOSITORY}/cp4waiops-bcdr:v4.2.1
      ```

   2. Run the following command to log in to the image registry:

      ```
      docker login ${IMAGE_REGISTRY} -u ${USERNAME}
      ```

   3. Run the following command to push your image to the image registry:

      ```
      docker push ${IMAGE_REGISTRY}/${REPOSITORY}/cp4waiops-bcdr:v4.2.1
      ```

6. If you need to run backup for IBM Cloud Pak for AIOps v4.2.1, follow the instructions in Backing up IBM Cloud Pak for AIOps data from step *Package the Helm Chart* onwards.

Set up restore on an upgraded IBM Cloud Pak for AIOps deployment

Use the following steps to upgrade your restore artifacts.

1. Uninstall the restore helm chart.

   Run the following command in the namespace that **Red Hat OpenShift APIs for Data Protection** (OADP) is installed in.

   ```
   helm uninstall restore-job
   ```

2. If your image registry server is unchanged since you previously set up restore, then skip this step. Otherwise, delete and create a new image pull secret.

   ```
   oc delete secret restore-secret -n ${VELERO_NAMESPACE}
   oc create secret docker-registry restore-secret -n ${VELERO_NAMESPACE} --docker-server=${IMAGE_REGISTRY}
   --docker-username=${USERNAME} --docker-password=${PASSWORD} --docker-email=${EMAIL}
   ```

3. Setup restore for IBM Cloud Pak for AIOps v4.2.1.

   If you need to run restore for IBM Cloud Pak for AIOps v4.2.1, follow the instructions in Restoring IBM Cloud Pak for AIOps to set up restore for IBM Cloud Pak for AIOps v4.2.1.

   Important:

   - The **cp4waiops-bcdr** docker image must be the same version as your IBM Cloud Pak for AIOps deployment, and must be the same **cp4waiops-bcdr** docker image that you created for setting up backup on your upgraded deployment. Follow these instructions to ensure that your **cp4waiops-bcdr** docker image is v4.2.1 for your IBM Cloud Pak for AIOps v4.2.1 deployment.
   - Skip the step that creates an image pull secret, as the required secret exists already.

Nonproduction features (Technology preview)

IBM Cloud Pak® for AIOps can include components and features for nonproduction use only as technology previews. As licensee of IBM Cloud Pak for AIOps, you can use these components and features only as part of your internal development and test environment for internal nonproduction use. Components and features that are available for nonproduction use are not to be relied on in a production environment.

Features available for nonproduction use:

- Application disruption costs (in AIOps insights)

  An application is assumed to be disrupted when one or more of its resources are alerting, and one or more of the associated alerts are included in a priority 1 (P1) incident. Disruption costs are calculated based on the amount of time that applications are in this state and the estimated disruption costs for the applications.

- Log anomaly automated training data selection

  A new toggle is available on the Filter data page when you are setting up training to enable or disable the ability to automatically select anomalous data. If you select *Auto select anomalies*, anomalies from the training data are filtered out to normalize the data source. This setting can helps AI distinguish between anomalous and normal behavior. For more information, see Setting up training for log anomaly detection - natural language.

- Incident Creation based on Persistence and Golden Signal

  Incident prioritization uses the golden signal policy and the alert suppression policy to filter out incidents when the observed alerts contain a golden signal and are not suppressed. The policy suppresses alerts that do not need attention.

- Feedback learning for statistical baseline log anomaly detection

  For some alerts, you can now select *Alert is valid* or *Alert is not valid* to help train a machine learning model. A message that corresponds to your selection appears in the alert details. For more information, see Alert details.

- Log anomaly model drift detection

  Incremental training prevents redundant training of preexisting models in consecutive training runs. With incremental training, each time that a training run launches, models are trained only on newly discovered resources and drifted resources, such as resources with changed log patterns. If neither of these resources are discovered in the training data since the previous training run, no new models are trained. Incremental training can significantly decrease training times. For more information, see *Enable or prevent incremental training* in Setting up training for log anomaly detection - natural language.

- About log anomaly detection - golden signals

  Detect log anomalies with golden signals. Golden signals help IT operations teams to classify and organize events when they diagnose incidents.

- Multi-zone high availability and disaster recovery (HADR)

  Install IBM Cloud Pak for AIOps on a multi-zone architecture (multi-zone HADR) for better resilience. IBM Cloud Pak for AIOps is starting to implement multi-zone availability, as provided by the Red Hat OpenShift Container Platform.

  Installing on a cluster that spans multiple availability zones improves resiliency by enabling the distribution of services across nodes in different zones in the cluster. If a zone becomes unavailable, then the remaining services in the other zones are still able to service requests, and disruption is lessened.

- Datadog integration for event and topology data collection

  A new observer and probe integration for connecting to Datadog is available.

  - Datadog Observer jobs

    With This observer, you can load monitored hosts and their associated processes, and then visualize this data as a topology view in the IBM Cloud Pak for AIOps UI.

  - Datadog integration probe

    With this probe integration, IBM Cloud Pak for AIOps sends HTTP GET requests to Datadog to retrieve events based on the time frame set by start and end mandatory query parameters in the URI.

APIs

IBM Cloud Pak® for AIOps offers a set of tools and APIs that you can use to predict, communicate, and resolve events before they become serious problems for your organization.

Learn how to use the tools and APIs that work with IBM Cloud Pak for AIOps in the following topics:

- Custom resource definitions
- Accessing APIs
- Enabling metrics monitoring in API gateway Kong

- Enabling logging in API gateway Kong
- User management and Platform UI APIs
- User management (IAM) APIs
- licensing APIs
- Issue resolution API
- Probable Cause Customization API
- Change Risk API
- Metric API
- Automation (runbook) API
- Topology APIs
- Infrastructure Management API Guide
- Managed services APIs

Important: You can directly access the Swagger UI for only the Topology API and Metric API.

- For more information about accessing Metric service Swagger UI, see Accessing Metric service Swagger UI.
- For more information about accessing topology service Swagger UI, see Accessing Topology service Swagger UI.

Custom resource definitions

Use the following information to learn about the custom resource definitions (CRDs) used by IBM Cloud Pak for AIOps and Infrastructure Automation.

For more information about API conventions and definitions, see API Conventions in the Kubernetes documentation.

- IBM Cloud Pak for AIOps custom resource definitions
- Infrastructure Automation custom resource definitions

IBM Cloud Pak for AIOps custom resource definitions

Use this information to learn about the custom resource definitions (CRDs) used by IBM Cloud Pak for AIOps.

In the following table, CRDs that do not have a hyperlink must not be manually edited unless otherwise instructed, and are manipulated only by the operator.

Table 1. IBM Cloud Pak® for AIOps CRDs

Kind	API version	Operator name	Description
AIManagerMainProd	v1beta1	aimanager-operator	AIManagerMainProd is the Schema for the aimanagermainprods API. For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.
AIManager	v1beta1	aimanager-operator	AIManager Deploys the IBM AIOps AI Manager service components of IBM Cloud Pak for AIOps. For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.
Algorithm	v1alpha1	aimanager-operator	Algorithm is the Schema for the algorithms API
Algorithm	v1beta1	aimanager-operator	Algorithm is the Schema for the algorithms API
AIOpsEdge	v1beta1	aiopsedge-operator	AIOpsEdge is the Schema for the aiopsedges API
AIOpsKafkaTopic	v1beta1	aiopsedge-operator	AIOpsKafkaTopic is the Schema for the aiopskafkatopics API
AutomationAction	v1beta1	aiopsedge-operator	AutomationAction is the Schema for the automationactions API
BundleManifest	v1alpha1	aiopsedge-operator	BundleManifest is the Schema for the bundlemanifests API
BundleManifest	v1beta1	aiopsedge-operator	BundleManifest is the Schema for the bundlemanifests API
ConnectorComponent	v1beta1	aiopsedge-operator	ConnectorComponent is the Schema for the connectorcomponents API
ConnectorConfiguration	v1beta1	aiopsedge-operator	ConnectorConfiguration is the Schema for the connectorconfigurations API

Kind	API version	Operator name	Description
ConnectorSchema	v1beta1	aiopsedge-operator	ConnectorSchema is the Schema for the connectorschemas API
GitApp	v1beta1	aiopsedge-operator	GitApp is the Schema for the GitApp API
MicroEdgeConfiguration	v1beta1	aiopsedge-operator	MicroEdgeConfiguration is the Schema for the microedgeconfigurations API
ASMFormation	v1beta1	asm-operator	'Create a formation deployment. Documentation This product is not deployable standalone and it is part of the LICCR already accepted'
ASM	v1beta1	asm-operator	Create Agile Service Manager deployment. Documentation for additional details regarding install parameters check: Learn More. By installing this product you accept: License Terms
AIOpsAnalyticsOrchestrator	v1beta1	ibm-aiops-ir-ai	AIOpsAnalyticsOrchestrator is the Schema for the aiopsanalyticsorchestrators API. This product is not deployable standalone and it is part of the LICCR already accepted
IssueResolutionCore	v1beta1	ibm-aiops-ir-core	This product is not deployable standalone and it is part of the LICCR already accepted.
LifecycleService	v1beta1	ibm-aiops-ir-lifecycle	LifecycleService is the Schema for the lifecycleservices API
LifecycleTrigger	v1beta1	ibm-aiops-ir-lifecycle	LifecycleTrigger is the Schema for the lifecycletriggers API
CustomSize	v1alpha1	ibm-aiops-orchestrator	CustomSize is the Schema for the customsizes API. This API is an alpha feature that has not been made generally available.
Installation	v1alpha1	ibm-aiops-orchestrator	Leverage advanced, explainable AI to resolve incidents faster, gain visibility into critical workloads, and intelligently manage key application resources. Learn More. By installing this product you accept the license terms.
Kong	v1alpha1	ibm-management-kong	Kong is the Schema for kong gateway
Tunnel	v1	ibm-secure-tunnel-operator	Tunnel is the Schema for the tunnels API
AIModelUI	v1	ibm-watson-aiops-ui-operator	For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.
AIOpsUIExtension	v1	ibm-watson-aiops-ui-operator	AIOpsUIExtension is the Schema for the aiopsuiextensions API
AIOpsUI	v1	ibm-watson-aiops-ui-operator	For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms. AIOpsUI is the Schema for the aiopsuis API.
BaseUI	v1	ibm-watson-aiops-ui-operator	For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.
ConnectorUI	v1	ibm-watson-aiops-ui-operator	For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.
InsightsUI	v1	ibm-watson-aiops-ui-operator	For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.

AIModelUI/v1

For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.

spec

Description: CommonSpecs - specs that are common to determining a deployment.

Type: object

Table 1. Description of AIModelUI/v1 spec

Property	Type	Description
spec	object	CommonSpecs - specs that are common to determining a deployment.

Property	Type	Description
spec.configuration	object	Additional configuration for the UI deployment
spec.configuration.topologyModel	string	TopologyModel - Specify the terminology to use in IBM Cloud Pak for AIOps when defining collections of topology resource groups. Valid values are "application" or "service". If this setting is not specified, the default value will be "application".
spec.container	object	ContainerSpec - possible configurations around containers
spec.container.containerPort	integer	(No Description)
spec.container.env	array	(No Description)
spec.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.image	object	ImageSpec - specification around how to pull an image.
spec.image.path	string	(No Description)
spec.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.image.pullSecret	string	(No Description)
spec.license	object	License - license spec determines whether a license has been accepted
spec.license.accept	boolean	License acceptance switch
spec.name	string	Name of the service being deployed
spec.replicas	integer	(No Description)
spec.size	string	(No Description)
spec.tlsSecretName	string	(No Description)
spec.version	string	Version of the csv deploying
spec.zenserviceName	string	name of the zenservice cr that is used for this deployment

status

Description: (No Description)

Type: object

Table 1. Description of AIModelUI/v1 status

Property	Type	Description
status	object	(No Description)
status.conditions	array	(No Description)
status.managedResources	array	(No Description)
status.phase	string	(No Description)
status.versions	object	(No Description)
status.versions.reconciled	string	(No Description)

AIOpsEdge/v1beta1

AIOpsEdge is the Schema for the aiopsedges API

spec

Description: AIOpsEdgeSpec defines the desired state of AIOpsEdge

Type: object

Table 1. Description of AIOpsEdge/v1beta1 spec

Property	Type	Description
spec	object	AIOpsEdgeSpec defines the desired state of AIOpsEdge
spec.disableDefaultConnectors	boolean	DisableDefaultConnectors can be enabled to prevent the installation of default connectors
spec.kafka	object	The specification of the Kafka and ZooKeeper clusters, and Topic Operator.
spec.kafka.clientsCa	object	Configuration of the clients certificate authority.
spec.kafka.clientsCa.certificateExpirationPolicy	string	How should CA certificate expiration be handled when `generateCertificateAuthority=true`. The default is for a new CA certificate to be generated reusing the existing private key.
spec.kafka.clientsCa.generateCertificateAuthority	boolean	If true then Certificate Authority certificates will be generated automatically. Otherwise the user will need to provide a Secret with the CA certificate. Default is true.
spec.kafka.clientsCa.generateSecretOwnerReference	boolean	If true then the Certificate Authority certificates secrets owner reference is set to the Kafka object. Otherwise no owner reference is set so deleting the Kafka object won't delete the secrets. Default is true.
spec.kafka.clientsCa.renewalDays	integer	The number of days in the certificate renewal period. This is the number of days before the a certificate expires during which renewal actions may be performed. When `generateCertificateAuthority` is true, this will cause the generation of a new certificate. When `generateCertificateAuthority` is true, this will cause extra logging at WARN level about the pending certificate expiry. Default is 30.
spec.kafka.clientsCa.validityDays	integer	The number of days generated certificates should be valid for. The default is 365.
spec.kafka.clusterCa	object	Configuration of the cluster certificate authority.
spec.kafka.clusterCa.certificateExpirationPolicy	string	How should CA certificate expiration be handled when `generateCertificateAuthority=true`. The default is for a new CA certificate to be generated reusing the existing private key.
spec.kafka.clusterCa.generateCertificateAuthority	boolean	If true then Certificate Authority certificates will be generated automatically. Otherwise the user will need to provide a Secret with the CA certificate. Default is true.
spec.kafka.clusterCa.generateSecretOwnerReference	boolean	If true then the Certificate Authority certificates secrets owner reference is set to the Kafka object. Otherwise no owner reference is set so deleting the Kafka object won't delete the secrets. Default is true.
spec.kafka.clusterCa.renewalDays	integer	The number of days in the certificate renewal period. This is the number of days before the a certificate expires during which renewal actions may be performed. When `generateCertificateAuthority` is true, this will cause the generation of a new certificate. When `generateCertificateAuthority` is true, this will cause extra logging at WARN level about the pending certificate expiry. Default is 30.
spec.kafka.clusterCa.validityDays	integer	The number of days generated certificates should be valid for. The default is 365.
spec.kafka.cruiseControl	object	Configuration for Cruise Control deployment. Deploys a Cruise Control instance when specified.
spec.kafka.cruiseControl.brokerCapacity	object	The Cruise Control `brokerCapacity` configuration.
spec.kafka.cruiseControl.brokerCapacity.cpuUtilization	integer	Broker capacity for CPU resource utilization as a percentage (0 - 100).
spec.kafka.cruiseControl.brokerCapacity.disk	string	Broker capacity for disk in bytes, for example, 100Gi.
spec.kafka.cruiseControl.brokerCapacity.inboundNetwork	string	Broker capacity for inbound network throughput in bytes per second, for example, 10000KB/s.
spec.kafka.cruiseControl.brokerCapacity.outboundNetwork	string	Broker capacity for outbound network throughput in bytes per second, for example 10000KB/s.

Property	Type	Description
spec.kafka.cruiseControl.config	object	The Cruise Control configuration. For a full list of configuration options refer to https://github.com/linkedin/cruise-control/wiki/Configurations. Note that properties with the following prefixes cannot be set: bootstrap.servers, client.id, zookeeper., network., security., failed.brokers.zk.path,webserver.http., webserver.api.urlprefix, webserver.session.path, webserver.accesslog., two.step., request.reason.required,metric.reporter.sampler.bootstrap.servers, metric.reporter.topic, partition.metric.sample.store.topic, broker.metric.sample.store.topic,capacity.config.file, self.healing., anomaly.detection., ssl. (with the exception of: ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols, webserver.http.cors.enabled,webserver.http.cors.origin, webserver.http.cors.exposeheaders).
spec.kafka.cruiseControl.image	string	The docker image for the pods.
spec.kafka.cruiseControl.jvmOptions	object	JVM Options for the Cruise Control container.
spec.kafka.cruiseControl.jvmOptions.-XX	object	A map of -XX options to the JVM.
spec.kafka.cruiseControl.jvmOptions.-Xms	string	-Xms option to to the JVM.
spec.kafka.cruiseControl.jvmOptions.-Xmx	string	-Xmx option to to the JVM.
spec.kafka.cruiseControl.jvmOptions.gcLoggingEnabled	boolean	Specifies whether the Garbage Collection logging is enabled. The default is false.
spec.kafka.cruiseControl.jvmOptions.javaSystemProperties	array	A map of additional system properties which will be passed using the `-D` option to the JVM.
spec.kafka.cruiseControl.livenessProbe	object	Pod liveness checking for the Cruise Control container.
spec.kafka.cruiseControl.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.cruiseControl.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.cruiseControl.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.cruiseControl.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.cruiseControl.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.cruiseControl.logging	object	Logging configuration (log4j1) for Cruise Control.
spec.kafka.cruiseControl.logging.loggers	object	A Map from logger name to logger level.
spec.kafka.cruiseControl.logging.name	string	The name of the `ConfigMap` from which to get the logging configuration.
spec.kafka.cruiseControl.logging.type	string	Logging type, must be either 'inline' or 'external'.
spec.kafka.cruiseControl.metrics	object	The Prometheus JMX Exporter configuration. See https://github.com/prometheus/jmx_exporter for details of the structure of this configuration.

Property	Type	Description
spec.kafka.cruiseControl.metricsConfig	object	Metrics configuration.
spec.kafka.cruiseControl.metricsConfig.type	string	Metrics type. Only 'jmxPrometheusExporter' supported currently.
spec.kafka.cruiseControl.metricsConfig.valueFrom	object	ConfigMap where the Prometheus JMX Exporter configuration is stored. For details of the structure of this configuration, see the {JMXExporter}.
spec.kafka.cruiseControl.metricsConfig.valueFrom.configMapKeyRef	object	Reference to the key in the ConfigMap containing the metrics configuration.
spec.kafka.cruiseControl.metricsConfig.valueFrom.configMapKeyRef.key	string	Key corresponds to the JSON schema field "key".
spec.kafka.cruiseControl.metricsConfig.valueFrom.configMapKeyRef.name	string	Name corresponds to the JSON schema field "name".
spec.kafka.cruiseControl.metricsConfig.valueFrom.configMapKeyRef.optional	boolean	Optional corresponds to the JSON schema field "optional".
spec.kafka.cruiseControl.readinessProbe	object	Pod readiness checking for the Cruise Control container.
spec.kafka.cruiseControl.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.cruiseControl.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.cruiseControl.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.cruiseControl.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.cruiseControl.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.cruiseControl.resources	object	CPU and memory resources to reserve for the Cruise Control container.
spec.kafka.cruiseControl.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.cruiseControl.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.cruiseControl.template	object	Template to specify how Cruise Control resources, `Deployments` and `Pods`, are generated.
spec.kafka.cruiseControl.template.apiService	object	Template for Cruise Control API `Service`.

Property	Type	Description
spec.kafka.cruiseControl.template.apiService.metadata	object	Metadata applied to the resource.
spec.kafka.cruiseControl.template.apiService.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.cruiseControl.template.apiService.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.cruiseControl.template.cruiseControlContainer	object	Template for the Cruise Control container.
spec.kafka.cruiseControl.template.cruiseControlContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext	object	Security context for the container.
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".

Property	Type	Description
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".

Property	Type	Description
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.cruiseControl.template.cruiseControlContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.cruiseControl.template.deployment	object	Template for Cruise Control `Deployment`.
spec.kafka.cruiseControl.template.deployment.metadata	object	Metadata applied to the resource.
spec.kafka.cruiseControl.template.deployment.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.cruiseControl.template.deployment.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.cruiseControl.template.pod	object	Template for Cruise Control `Pods`.
spec.kafka.cruiseControl.template.pod.affinity	object	The pod's affinity rules.
spec.kafka.cruiseControl.template.pod.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.cruiseControl.template.pod.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.cruiseControl.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".

Property	Type	Description
spec.kafka.cruiseControl.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.cruiseControl.template.pod.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.cruiseControl.template.pod.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.cruiseControl.template.pod.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.cruiseControl.template.pod.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.cruiseControl.template.pod.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.cruiseControl.template.pod.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.cruiseControl.template.pod.hostAliases	array	The pod's HostAliases. HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified.
spec.kafka.cruiseControl.template.pod.imagePullSecrets	array	List of references to secrets in the same namespace to use for pulling any of the images used by this Pod. When the **`STRIMZI_IMAGE_PULL_SECRETS`** environment variable in Cluster Operator and the **`imagePullSecrets`** option are specified, only the **`imagePullSecrets`** variable is used and the **`STRIMZI_IMAGE_PULL_SECRETS`** variable is ignored.
spec.kafka.cruiseControl.template.pod.metadata	object	Metadata applied to the resource.
spec.kafka.cruiseControl.template.pod.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.cruiseControl.template.pod.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.cruiseControl.template.pod.priorityClassName	string	The name of the priority class used to assign priority to the pods. For more information about priority classes, see {K8sPriorityClass}.

Property	Type	Description
spec.kafka.cruiseControl.template.pod.schedulerName	string	The name of the scheduler used to dispatch this `Pod`. If not specified, the default scheduler will be used.
spec.kafka.cruiseControl.template.pod.securityContext	object	Configures pod-level security attributes and common container settings.
spec.kafka.cruiseControl.template.pod.securityContext.fsGroup	integer	FsGroup corresponds to the JSON schema field "fsGroup".
spec.kafka.cruiseControl.template.pod.securityContext.fsGroupChangePolicy	string	FsGroupChangePolicy corresponds to the JSON schema field "fsGroupChangePolicy".
spec.kafka.cruiseControl.template.pod.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.cruiseControl.template.pod.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.cruiseControl.template.pod.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.cruiseControl.template.pod.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.cruiseControl.template.pod.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.cruiseControl.template.pod.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.cruiseControl.template.pod.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.cruiseControl.template.pod.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.cruiseControl.template.pod.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.cruiseControl.template.pod.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".

Property	Type	Description
spec.kafka.cruiseControl.template.pod.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.cruiseControl.template.pod.securityContext.supplementalGroups	array	SupplementalGroups corresponds to the JSON schema field "supplementalGroups".
spec.kafka.cruiseControl.template.pod.securityContext.sysctls	array	Sysctls corresponds to the JSON schema field "sysctls".
spec.kafka.cruiseControl.template.pod.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.cruiseControl.template.pod.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.cruiseControl.template.pod.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.cruiseControl.template.pod.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.cruiseControl.template.pod.terminationGracePeriodSeconds	integer	The grace period is the duration in seconds after the processes running in the pod are sent a termination signal, and the time when the processes are forcibly halted with a kill signal. Set this value to longer than the expected cleanup time for your process. Value must be a non-negative integer. A zero value indicates delete immediately. You might need to increase the grace period for very large Kafka clusters, so that the Kafka brokers have enough time to transfer their work to another broker before they are terminated. Defaults to 30 seconds.
spec.kafka.cruiseControl.template.pod.tolerations	array	The pod's tolerations.
spec.kafka.cruiseControl.template.pod.topologySpreadConstraints	array	The pod's topology spread constraints.
spec.kafka.cruiseControl.template.podDisruptionBudget	object	Template for Cruise Control **`PodDisruptionBudget`**.
spec.kafka.cruiseControl.template.podDisruptionBudget.maxUnavailable	integer	Maximum number of unavailable pods to allow automatic Pod eviction. A Pod eviction is allowed when the **`maxUnavailable`** number of pods or fewer are unavailable after the eviction. Setting this value to 0 prevents all voluntary evictions, so the pods must be evicted manually. Defaults to 1.
spec.kafka.cruiseControl.template.podDisruptionBudget.metadata	object	Metadata to apply to the **`PodDistruptionBugetTemplate`** resource.
spec.kafka.cruiseControl.template.podDisruptionBudget.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.

Property	Type	Description
spec.kafka.cruiseControl.template.podDisruptionBudget.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.cruiseControl.template.tlsSidecarContainer	object	Template for the Cruise Control TLS sidecar container.
spec.kafka.cruiseControl.template.tlsSidecarContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext	object	Security context for the container.
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".

Property	Type	Description
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".

Property	Type	Description
spec.kafka.cruiseControl.template.tlsSidecarContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.cruiseControl.tlsSidecar	object	TLS sidecar configuration.
spec.kafka.cruiseControl.tlsSidecar.image	string	The docker image for the container.
spec.kafka.cruiseControl.tlsSidecar.livenessProbe	object	Pod liveness checking.
spec.kafka.cruiseControl.tlsSidecar.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.cruiseControl.tlsSidecar.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.cruiseControl.tlsSidecar.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.cruiseControl.tlsSidecar.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.cruiseControl.tlsSidecar.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.cruiseControl.tlsSidecar.logLevel	string	The log level for the TLS sidecar. Default value is **`notice`**.
spec.kafka.cruiseControl.tlsSidecar.readinessProbe	object	Pod readiness checking.
spec.kafka.cruiseControl.tlsSidecar.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.cruiseControl.tlsSidecar.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.cruiseControl.tlsSidecar.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.cruiseControl.tlsSidecar.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.cruiseControl.tlsSidecar.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.cruiseControl.tlsSidecar.resources	object	CPU and memory resources to reserve.

Property	Type	Description
spec.kafka.cruiseControl.tlsSidecar.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.cruiseControl.tlsSidecar.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.entityOperator	object	Configuration of the Entity Operator.
spec.kafka.entityOperator.affinity	object	The pod's affinity rules.
spec.kafka.entityOperator.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.entityOperator.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.entityOperator.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.entityOperator.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.entityOperator.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.template	object	Template for Entity Operator resources. The template allows users to specify how is the **`Deployment`** and **`Pods`** generated.

Property	Type	Description
spec.kafka.entityOperator.template.deployment	object	Template for Entity Operator `Deployment`.
spec.kafka.entityOperator.template.deployment.metadata	object	Metadata applied to the resource.
spec.kafka.entityOperator.template.deployment.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.entityOperator.template.deployment.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.entityOperator.template.pod	object	Template for Entity Operator `Pods`.
spec.kafka.entityOperator.template.pod.affinity	object	The pod's affinity rules.
spec.kafka.entityOperator.template.pod.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.entityOperator.template.pod.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.entityOperator.template.pod.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.entityOperator.template.pod.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.template.pod.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".

Property	Type	Description
spec.kafka.entityOperator.template.pod.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.entityOperator.template.pod.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.template.pod.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.entityOperator.template.pod.hostAliases	array	The pod's HostAliases. HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified.
spec.kafka.entityOperator.template.pod.imagePullSecrets	array	List of references to secrets in the same namespace to use for pulling any of the images used by this Pod. When the **`STRIMZI_IMAGE_PULL_SECRETS`** environment variable in Cluster Operator and the **`imagePullSecrets`** option are specified, only the **`imagePullSecrets`** variable is used and the **`STRIMZI_IMAGE_PULL_SECRETS`** variable is ignored.
spec.kafka.entityOperator.template.pod.metadata	object	Metadata applied to the resource.
spec.kafka.entityOperator.template.pod.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.entityOperator.template.pod.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.entityOperator.template.pod.priorityClassName	string	The name of the priority class used to assign priority to the pods. For more information about priority classes, see {K8sPriorityClass}.
spec.kafka.entityOperator.template.pod.schedulerName	string	The name of the scheduler used to dispatch this **`Pod`**. If not specified, the default scheduler will be used.
spec.kafka.entityOperator.template.pod.securityContext	object	Configures pod-level security attributes and common container settings.
spec.kafka.entityOperator.template.pod.securityContext.fsGroup	integer	FsGroup corresponds to the JSON schema field "fsGroup".
spec.kafka.entityOperator.template.pod.securityContext.fsGroupChangePolicy	string	FsGroupChangePolicy corresponds to the JSON schema field "fsGroupChangePolicy".
spec.kafka.entityOperator.template.pod.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.entityOperator.template.pod.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".

Property	Type	Description
spec.kafka.entityOperator.template.pod.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.entityOperator.template.pod.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.entityOperator.template.pod.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.entityOperator.template.pod.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.entityOperator.template.pod.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.entityOperator.template.pod.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.entityOperator.template.pod.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.entityOperator.template.pod.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.entityOperator.template.pod.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.entityOperator.template.pod.securityContext.supplementalGroups	array	SupplementalGroups corresponds to the JSON schema field "supplementalGroups".
spec.kafka.entityOperator.template.pod.securityContext.sysctls	array	Sysctls corresponds to the JSON schema field "sysctls".
spec.kafka.entityOperator.template.pod.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.entityOperator.template.pod.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".

Property	Type	Description
spec.kafka.entityOperator.template.pod.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.entityOperator.template.pod.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.entityOperator.template.pod.terminationGracePeriodSeconds	integer	The grace period is the duration in seconds after the processes running in the pod are sent a termination signal, and the time when the processes are forcibly halted with a kill signal. Set this value to longer than the expected cleanup time for your process. Value must be a non-negative integer. A zero value indicates delete immediately. You might need to increase the grace period for very large Kafka clusters, so that the Kafka brokers have enough time to transfer their work to another broker before they are terminated. Defaults to 30 seconds.
spec.kafka.entityOperator.template.pod.tolerations	array	The pod's tolerations.
spec.kafka.entityOperator.template.pod.topologySpreadConstraints	array	The pod's topology spread constraints.
spec.kafka.entityOperator.template.tlsSidecarContainer	object	Template for the Entity Operator TLS sidecar container.
spec.kafka.entityOperator.template.tlsSidecarContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext	object	Security context for the container.
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".

Property	Type	Description
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".

Property	Type	Description
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.entityOperator.template.tlsSidecarContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.entityOperator.template.topicOperatorContainer	object	Template for the Entity Topic Operator container.
spec.kafka.entityOperator.template.topicOperatorContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext	object	Security context for the container.
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".

Property	Type	Description
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".

Property	Type	Description
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.entityOperator.template.topicOperatorContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.entityOperator.template.userOperatorContainer	object	Template for the Entity User Operator container.
spec.kafka.entityOperator.template.userOperatorContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.entityOperator.template.userOperatorContainer.securityContext	object	Security context for the container.
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".

Property	Type	Description
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".

Property	Type	Description
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.entityOperator.template.userOperatorContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.entityOperator.tlsSidecar	object	TLS sidecar configuration.
spec.kafka.entityOperator.tlsSidecar.image	string	The docker image for the container.
spec.kafka.entityOperator.tlsSidecar.livenessProbe	object	Pod liveness checking.

Property	Type	Description
spec.kafka.entityOperator.tlsSidecar.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.entityOperator.tlsSidecar.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.entityOperator.tlsSidecar.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.entityOperator.tlsSidecar.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.entityOperator.tlsSidecar.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.entityOperator.tlsSidecar.logLevel	string	The log level for the TLS sidecar. Default value is `notice`.
spec.kafka.entityOperator.tlsSidecar.readinessProbe	object	Pod readiness checking.
spec.kafka.entityOperator.tlsSidecar.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.entityOperator.tlsSidecar.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.entityOperator.tlsSidecar.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.entityOperator.tlsSidecar.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.entityOperator.tlsSidecar.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.entityOperator.tlsSidecar.resources	object	CPU and memory resources to reserve.
spec.kafka.entityOperator.tlsSidecar.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.entityOperator.tlsSidecar.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.entityOperator.tolerations	array	The pod's tolerations.
spec.kafka.entityOperator.topicOperator	object	Configuration of the Topic Operator.
spec.kafka.entityOperator.topicOperator.image	string	The image to use for the Topic Operator.

Property	Type	Description
spec.kafka.entityOperator.topicOperator.jvmOptions	object	JVM Options for pods.
spec.kafka.entityOperator.topicOperator.jvmOptions.-XX	object	A map of -XX options to the JVM.
spec.kafka.entityOperator.topicOperator.jvmOptions.-Xms	string	-Xms option to to the JVM.
spec.kafka.entityOperator.topicOperator.jvmOptions.-Xmx	string	-Xmx option to to the JVM.
spec.kafka.entityOperator.topicOperator.jvmOptions.gcLoggingEnabled	boolean	Specifies whether the Garbage Collection logging is enabled. The default is false.
spec.kafka.entityOperator.topicOperator.jvmOptions.javaSystemProperties	array	A map of additional system properties which will be passed using the `-D` option to the JVM.
spec.kafka.entityOperator.topicOperator.livenessProbe	object	Pod liveness checking.
spec.kafka.entityOperator.topicOperator.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.entityOperator.topicOperator.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.entityOperator.topicOperator.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.entityOperator.topicOperator.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.entityOperator.topicOperator.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.entityOperator.topicOperator.logging	object	Logging configuration.
spec.kafka.entityOperator.topicOperator.logging.loggers	object	A Map from logger name to logger level.
spec.kafka.entityOperator.topicOperator.logging.name	string	The name of the `ConfigMap` from which to get the logging configuration.
spec.kafka.entityOperator.topicOperator.logging.type	string	Logging type, must be either 'inline' or 'external'.
spec.kafka.entityOperator.topicOperator.readinessProbe	object	Pod readiness checking.

Property	Type	Description
spec.kafka.entityOperator.topicOperator.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.entityOperator.topicOperator.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.entityOperator.topicOperator.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.entityOperator.topicOperator.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.entityOperator.topicOperator.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.entityOperator.topicOperator.reconciliationIntervalSeconds	integer	Interval between periodic reconciliations.
spec.kafka.entityOperator.topicOperator.resources	object	CPU and memory resources to reserve.
spec.kafka.entityOperator.topicOperator.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.entityOperator.topicOperator.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.entityOperator.topicOperator.topicMetadataMaxAttempts	integer	The number of attempts at getting topic metadata.
spec.kafka.entityOperator.topicOperator.watchedNamespace	string	The namespace the Topic Operator should watch.
spec.kafka.entityOperator.topicOperator.zookeeperSessionTimeoutSeconds	integer	Timeout for the ZooKeeper session.
spec.kafka.entityOperator.userOperator	object	Configuration of the User Operator.
spec.kafka.entityOperator.userOperator.image	string	The image to use for the User Operator.
spec.kafka.entityOperator.userOperator.jvmOptions	object	JVM Options for pods.
spec.kafka.entityOperator.userOperator.jvmOptions.-XX	object	A map of -XX options to the JVM.

Property	Type	Description
spec.kafka.entityOperator.userOperator.jvmOptions.-Xms	string	-Xms option to to the JVM.
spec.kafka.entityOperator.userOperator.jvmOptions.-Xmx	string	-Xmx option to to the JVM.
spec.kafka.entityOperator.userOperator.jvmOptions.gcLoggingEnabled	boolean	Specifies whether the Garbage Collection logging is enabled. The default is false.
spec.kafka.entityOperator.userOperator.jvmOptions.javaSystemProperties	array	A map of additional system properties which will be passed using the `-D` option to the JVM.
spec.kafka.entityOperator.userOperator.livenessProbe	object	Pod liveness checking.
spec.kafka.entityOperator.userOperator.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.entityOperator.userOperator.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.entityOperator.userOperator.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.entityOperator.userOperator.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.entityOperator.userOperator.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.entityOperator.userOperator.logging	object	Logging configuration.
spec.kafka.entityOperator.userOperator.logging.loggers	object	A Map from logger name to logger level.
spec.kafka.entityOperator.userOperator.logging.name	string	The name of the `ConfigMap` from which to get the logging configuration.
spec.kafka.entityOperator.userOperator.logging.type	string	Logging type, must be either 'inline' or 'external'.
spec.kafka.entityOperator.userOperator.readinessProbe	object	Pod readiness checking.
spec.kafka.entityOperator.userOperator.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.

Property	Type	Description
spec.kafka.entityOperator.userOperator.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.entityOperator.userOperator.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.entityOperator.userOperator.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.entityOperator.userOperator.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.entityOperator.userOperator.reconciliationIntervalSeconds	integer	Interval between periodic reconciliations.
spec.kafka.entityOperator.userOperator.resources	object	CPU and memory resources to reserve.
spec.kafka.entityOperator.userOperator.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.entityOperator.userOperator.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.entityOperator.userOperator.secretPrefix	string	The prefix that will be added to the KafkaUser name to be used as the Secret name.
spec.kafka.entityOperator.userOperator.watchedNamespace	string	The namespace the User Operator should watch.
spec.kafka.entityOperator.userOperator.zookeeperSessionTimeoutSeconds	integer	Timeout for the ZooKeeper session.
spec.kafka.jmxTrans	object	Configuration for JmxTrans. When the property is present a JmxTrans deployment is created for gathering JMX metrics from each Kafka broker. For more information see https://github.com/jmxtrans/jmxtrans[JmxTrans GitHub].
spec.kafka.jmxTrans.image	string	The image to use for the JmxTrans.
spec.kafka.jmxTrans.kafkaQueries	array	Queries to send to the Kafka brokers to define what data should be read from each broker. For more information on these properties see, xref:type-JmxTransQueryTemplate-reference[`JmxTransQueryTemplate` schema reference].
spec.kafka.jmxTrans.logLevel	string	Sets the logging level of the JmxTrans deployment.For more information see, https://github.com/jmxtrans/jmxtrans-agent/wiki/Troubleshooting[JmxTrans Logging Level].
spec.kafka.jmxTrans.outputDefinitions	array	Defines the output hosts that will be referenced later on. For more information on these properties see, xref:type-JmxTransOutputDefinitionTemplate-reference[`JmxTransOutputDefinitionTemplate` schema reference].
spec.kafka.jmxTrans.resources	object	CPU and memory resources to reserve.
spec.kafka.jmxTrans.resources.limits	object	Limits corresponds to the JSON schema field "limits".

Property	Type	Description
spec.kafka.jmxTrans.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.jmxTrans.template	object	Template for JmxTrans resources.
spec.kafka.jmxTrans.template.container	object	Template for JmxTrans container.
spec.kafka.jmxTrans.template.container.env	array	Environment variables which should be applied to the container.
spec.kafka.jmxTrans.template.container.securityContext	object	Security context for the container.
spec.kafka.jmxTrans.template.container.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.jmxTrans.template.container.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.jmxTrans.template.container.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.jmxTrans.template.container.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.jmxTrans.template.container.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.jmxTrans.template.container.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.jmxTrans.template.container.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.jmxTrans.template.container.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.jmxTrans.template.container.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.jmxTrans.template.container.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.jmxTrans.template.container.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".

Property	Type	Description
spec.kafka.jmxTrans.template.container.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.jmxTrans.template.container.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.jmxTrans.template.container.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.jmxTrans.template.container.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.jmxTrans.template.container.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.jmxTrans.template.container.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.jmxTrans.template.container.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.jmxTrans.template.container.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.jmxTrans.template.container.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.jmxTrans.template.container.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.jmxTrans.template.container.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.jmxTrans.template.deployment	object	Template for JmxTrans **`Deployment`**.
spec.kafka.jmxTrans.template.deployment.metadata	object	Metadata applied to the resource.

Property	Type	Description
spec.kafka.jmxTrans.template.deployment.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.jmxTrans.template.deployment.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.jmxTrans.template.pod	object	Template for JmxTrans `Pods`.
spec.kafka.jmxTrans.template.pod.affinity	object	The pod's affinity rules.
spec.kafka.jmxTrans.template.pod.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.jmxTrans.template.pod.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.jmxTrans.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.jmxTrans.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.jmxTrans.template.pod.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.jmxTrans.template.pod.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.jmxTrans.template.pod.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.jmxTrans.template.pod.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.jmxTrans.template.pod.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".

Property	Type	Description
spec.kafka.jmxTrans.template.pod.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.jmxTrans.template.pod.hostAliases	array	The pod's HostAliases. HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified.
spec.kafka.jmxTrans.template.pod.imagePullSecrets	array	List of references to secrets in the same namespace to use for pulling any of the images used by this Pod. When the **`STRIMZI_IMAGE_PULL_SECRETS`** environment variable in Cluster Operator and the **`imagePullSecrets`** option are specified, only the **`imagePullSecrets`** variable is used and the **`STRIMZI_IMAGE_PULL_SECRETS`** variable is ignored.
spec.kafka.jmxTrans.template.pod.metadata	object	Metadata applied to the resource.
spec.kafka.jmxTrans.template.pod.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.jmxTrans.template.pod.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.jmxTrans.template.pod.priorityClassName	string	The name of the priority class used to assign priority to the pods. For more information about priority classes, see {K8sPriorityClass}.
spec.kafka.jmxTrans.template.pod.schedulerName	string	The name of the scheduler used to dispatch this **`Pod`**. If not specified, the default scheduler will be used.
spec.kafka.jmxTrans.template.pod.securityContext	object	Configures pod-level security attributes and common container settings.
spec.kafka.jmxTrans.template.pod.securityContext.fsGroup	integer	FsGroup corresponds to the JSON schema field "fsGroup".
spec.kafka.jmxTrans.template.pod.securityContext.fsGroupChangePolicy	string	FsGroupChangePolicy corresponds to the JSON schema field "fsGroupChangePolicy".
spec.kafka.jmxTrans.template.pod.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.jmxTrans.template.pod.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.jmxTrans.template.pod.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.jmxTrans.template.pod.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.jmxTrans.template.pod.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.jmxTrans.template.pod.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".

Property	Type	Description
spec.kafka.jmxTrans.template.pod.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.jmxTrans.template.pod.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.jmxTrans.template.pod.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.jmxTrans.template.pod.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.jmxTrans.template.pod.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.jmxTrans.template.pod.securityContext.supplementalGroups	array	SupplementalGroups corresponds to the JSON schema field "supplementalGroups".
spec.kafka.jmxTrans.template.pod.securityContext.sysctls	array	Sysctls corresponds to the JSON schema field "sysctls".
spec.kafka.jmxTrans.template.pod.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.jmxTrans.template.pod.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.jmxTrans.template.pod.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.jmxTrans.template.pod.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.jmxTrans.template.pod.terminationGracePeriodSeconds	integer	The grace period is the duration in seconds after the processes running in the pod are sent a termination signal, and the time when the processes are forcibly halted with a kill signal. Set this value to longer than the expected cleanup time for your process. Value must be a non-negative integer. A zero value indicates delete immediately. You might need to increase the grace period for very large Kafka clusters, so that the Kafka brokers have enough time to transfer their work to another broker before they are terminated. Defaults to 30 seconds.
spec.kafka.jmxTrans.template.pod.tolerations	array	The pod's tolerations.
spec.kafka.jmxTrans.template.pod.topologySpreadConstraints	array	The pod's topology spread constraints.
spec.kafka.kafka	object	Configuration of the Kafka cluster.

Property	Type	Description
spec.kafka.kafka.affinity	object	The pod's affinity rules.
spec.kafka.kafka.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.kafka.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.kafka.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.kafka.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.kafka.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.authorization	object	Authorization configuration for Kafka brokers.
spec.kafka.kafka.authorization.allowOnError	boolean	Defines whether a Kafka client should be allowed or denied by default when the authorizer fails to query the Open Policy Agent, for example, when it is temporarily unavailable). Defaults to **`false`** - all actions will be denied.
spec.kafka.kafka.authorization.clientId	string	OAuth Client ID which the Kafka client can use to authenticate against the OAuth server and use the token endpoint URI.
spec.kafka.kafka.authorization.delegateToKafkaAcls	boolean	Whether authorization decision should be delegated to the 'Simple' authorizer if DENIED by Keycloak Authorization Services policies. Default value is **`false`**.
spec.kafka.kafka.authorization.disableTlsHostnameVerification	boolean	Enable or disable TLS hostname verification. Default value is **`false`**.
spec.kafka.kafka.authorization.expireAfterMs	integer	The expiration of the records kept in the local cache to avoid querying the Open Policy Agent for every request. Defines how often the cached authorization decisions are reloaded from the Open Policy Agent server. In milliseconds. Defaults to **`3600000`**.

Property	Type	Description
spec.kafka.kafka.authorization.grantsRefreshPeriodSeconds	integer	The time between two consecutive grants refresh runs in seconds. The default value is 60.
spec.kafka.kafka.authorization.grantsRefreshPoolSize	integer	The number of threads to use to refresh grants for active sessions. The more threads, the more parallelism, so the sooner the job completes. However, using more threads places a heavier load on the authorization server. The default value is 5.
spec.kafka.kafka.authorization.initialCacheCapacity	integer	Initial capacity of the local cache used by the authorizer to avoid querying the Open Policy Agent for every request Defaults to `5000`.
spec.kafka.kafka.authorization.maximumCacheSize	integer	Maximum capacity of the local cache used by the authorizer to avoid querying the Open Policy Agent for every request. Defaults to `50000`.
spec.kafka.kafka.authorization.superUsers	array	List of super users. Should contain list of user principals which should get unlimited access rights.
spec.kafka.kafka.authorization.tlsTrustedCertificates	array	Trusted certificates for TLS connection to the OAuth server.
spec.kafka.kafka.authorization.tokenEndpointUri	string	Authorization server token endpoint URI.
spec.kafka.kafka.authorization.type	string	Authorization type. Currently, the supported types are `simple`, `keycloak`, and `opa`. `simple` authorization type uses Kafka's `kafka.security.authorizer.AclAuthorizer` class for authorization. `keycloak` authorization type uses Keycloak Authorization Services for authorization. `opa` authorization type uses Open Policy Agent based authorization.
spec.kafka.kafka.authorization.url	string	The URL used to connect to the Open Policy Agent server. The URL has to include the policy which will be queried by the authorizer. This option is required.
spec.kafka.kafka.brokerRackInitImage	string	The image of the init container used for initializing the `broker.rack`.
spec.kafka.kafka.config	object	Kafka broker config properties with the following prefixes cannot be set: listeners, advertised., broker., listener., host.name, port, inter.broker.listener.name, sasl., ssl., security., password., principal.builder.class, log.dir, zookeeper.connect, zookeeper.set.acl, zookeeper.ssl, zookeeper.clientCnxnSocket, authorizer., super.user, cruise.control.metrics.topic, cruise.control.metrics.reporter.bootstrap.servers (with the exception of: zookeeper.connection.timeout.ms, ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols,cruise.control.metrics.topic.num.partitions, cruise.control.metrics.topic.replication.factor, cruise.control.metrics.topic.retention.ms,cruise.control.metrics.topic.auto.create.retries, cruise.control.metrics.topic.auto.create.timeout.ms,cruise.control.metrics.topic.min.insync.replicas).
spec.kafka.kafka.image	string	The docker image for the pods. The default value depends on the configured `Kafka.spec.kafka.version`.
spec.kafka.kafka.jmxOptions	object	JMX Options for Kafka brokers.
spec.kafka.kafka.jmxOptions.authentication	object	Authentication configuration for connecting to the Kafka JMX port.
spec.kafka.kafka.jmxOptions.authentication.type	string	Authentication type. Currently the only supported types are `password`.`password` type creates a username and protected port with no TLS.
spec.kafka.kafka.jvmOptions	object	JVM Options for pods.
spec.kafka.kafka.jvmOptions.-XX	object	A map of -XX options to the JVM.
spec.kafka.kafka.jvmOptions.-Xms	string	-Xms option to to the JVM.
spec.kafka.kafka.jvmOptions.-Xmx	string	-Xmx option to to the JVM.
spec.kafka.kafka.jvmOptions.gcLoggingEnabled	boolean	Specifies whether the Garbage Collection logging is enabled. The default is false.

Property	Type	Description
spec.kafka.kafka.jvmOptions.javaSystemProperties	array	A map of additional system properties which will be passed using the `-D` option to the JVM.
spec.kafka.kafka.listeners	array	Listeners corresponds to the JSON schema field "listeners".
spec.kafka.kafka.livenessProbe	object	Pod liveness checking.
spec.kafka.kafka.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.kafka.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.kafka.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.kafka.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.kafka.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.kafka.logging	object	Logging configuration for Kafka.
spec.kafka.kafka.logging.loggers	object	A Map from logger name to logger level.
spec.kafka.kafka.logging.name	string	The name of the `ConfigMap` from which to get the logging configuration.
spec.kafka.kafka.logging.type	string	Logging type, must be either 'inline' or 'external'.
spec.kafka.kafka.metrics	object	The Prometheus JMX Exporter configuration. See https://github.com/prometheus/jmx_exporter for details of the structure of this configuration.
spec.kafka.kafka.metricsConfig	object	Metrics configuration.
spec.kafka.kafka.metricsConfig.type	string	Metrics type. Only 'jmxPrometheusExporter' supported currently.
spec.kafka.kafka.metricsConfig.valueFrom	object	ConfigMap where the Prometheus JMX Exporter configuration is stored. For details of the structure of this configuration, see the {JMXExporter}.
spec.kafka.kafka.metricsConfig.valueFrom.configMapKeyRef	object	Reference to the key in the ConfigMap containing the metrics configuration.
spec.kafka.kafka.metricsConfig.valueFrom.configMapKeyRef.key	string	Key corresponds to the JSON schema field "key".
spec.kafka.kafka.metricsConfig.valueFrom.configMapKeyRef.name	string	Name corresponds to the JSON schema field "name".
spec.kafka.kafka.metricsConfig.valueFrom.configMapKeyRef.optional	boolean	Optional corresponds to the JSON schema field "optional".
spec.kafka.kafka.rack	object	Configuration of the `broker.rack` broker config.
spec.kafka.kafka.rack.topologyKey	string	A key that matches labels assigned to the Kubernetes cluster nodes. The value of the label is used to set the broker's `broker.rack` config and `client.rack` in Kafka Connect.
spec.kafka.kafka.readinessProbe	object	Pod readiness checking.

Property	Type	Description
spec.kafka.kafka.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.kafka.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.kafka.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.kafka.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.kafka.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.kafka.replicas	integer	The number of pods in the cluster.
spec.kafka.kafka.resources	object	CPU and memory resources to reserve.
spec.kafka.kafka.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.kafka.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.kafka.storage	object	Storage configuration (disk). Cannot be updated.
spec.kafka.kafka.storage.class	string	The storage class to use for dynamic volume allocation.
spec.kafka.kafka.storage.deleteClaim	boolean	Specifies if the persistent volume claim has to be deleted when the cluster is un-deployed.
spec.kafka.kafka.storage.id	integer	Storage identification number. It is mandatory only for storage volumes defined in a storage of type 'jbod'.
spec.kafka.kafka.storage.overrides	array	Overrides for individual brokers. The **`overrides`** field allows to specify a different configuration for different brokers.
spec.kafka.kafka.storage.selector	object	Specifies a specific persistent volume to use. It contains key:value pairs representing labels for selecting such a volume.
spec.kafka.kafka.storage.size	string	When type=persistent-claim, defines the size of the persistent volume claim (i.e 1Gi). Mandatory when type=persistent-claim.
spec.kafka.kafka.storage.sizeLimit	string	When type=ephemeral, defines the total amount of local storage required for this EmptyDir volume (for example 1Gi).
spec.kafka.kafka.storage.type	string	Storage type, must be either 'ephemeral', 'persistent-claim', or 'jbod'.
spec.kafka.kafka.storage.volumes	array	List of volumes as Storage objects representing the JBOD disks array.
spec.kafka.kafka.template	object	Template for Kafka cluster resources. The template allows users to specify how are the **`StatefulSet`**, **`Pods`** and **`Services`** generated.
spec.kafka.kafka.template.bootstrapService	object	Template for Kafka bootstrap **`Service`**.
spec.kafka.kafka.template.bootstrapService.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.bootstrapService.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.bootstrapService.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.

Property	Type	Description
spec.kafka.kafka.template.brokersService	object	Template for Kafka broker **`Service`**.
spec.kafka.kafka.template.brokersService.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.brokersService.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.brokersService.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.clusterCaCert	object	Template for Secret with Kafka Cluster certificate public key.
spec.kafka.kafka.template.clusterCaCert.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.clusterCaCert.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.clusterCaCert.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.externalBootstrapIngress	object	Template for Kafka external bootstrap **`Ingress`**.
spec.kafka.kafka.template.externalBootstrapIngress.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.externalBootstrapIngress.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.externalBootstrapIngress.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.externalBootstrapRoute	object	Template for Kafka external bootstrap **`Route`**.
spec.kafka.kafka.template.externalBootstrapRoute.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.externalBootstrapRoute.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.externalBootstrapRoute.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.externalBootstrapService	object	Template for Kafka external bootstrap **`Service`**.

Property	Type	Description
spec.kafka.kafka.template.externalBootstrapService.externalTrafficPolicy	string	Specifies whether the service routes external traffic to node-local or cluster-wide endpoints. `Cluster` may cause a second hop to another node and obscures the client source IP. `Local` avoids a second hop for LoadBalancer and Nodeport type services and preserves the client source IP (when supported by the infrastructure). If unspecified, Kubernetes will use `Cluster` as the default.
spec.kafka.kafka.template.externalBootstrapService.loadBalancerSourceRanges	array	A list of CIDR ranges (for example `10.0.0.0/8` or `130.211.204.1/32`) from which clients can connect to load balancer type listeners. If supported by the platform, traffic through the loadbalancer is restricted to the specified CIDR ranges. This field is applicable only for loadbalancer type services and is ignored if the cloud provider does not support the feature. For more information, see https://v1-17.docs.kubernetes.io/docs/tasks/access-application-cluster/configure-cloud-provider-firewall/.
spec.kafka.kafka.template.externalBootstrapService.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.externalBootstrapService.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafka.template.externalBootstrapService.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafka.template.initContainer	object	Template for the Kafka init container.
spec.kafka.kafka.template.initContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.kafka.template.initContainer.securityContext	object	Security context for the container.
spec.kafka.kafka.template.initContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.kafka.template.initContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.kafka.template.initContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.kafka.template.initContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.kafka.template.initContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.kafka.template.initContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.kafka.template.initContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".

Property	Type	Description
spec.kafka.kafka.template.initContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.kafka.template.initContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.kafka.template.initContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.kafka.template.initContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.kafka.template.initContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.kafka.template.initContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.kafka.template.initContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafka.template.initContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.kafka.template.initContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.kafka.template.initContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.kafka.template.initContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafka.template.initContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.kafka.template.initContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".

Property	Type	Description
spec.kafka.kafka.template.initContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.kafka.template.initContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.kafka.template.kafkaContainer	object	Template for the Kafka broker container.
spec.kafka.kafka.template.kafkaContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.kafka.template.kafkaContainer.securityContext	object	Security context for the container.
spec.kafka.kafka.template.kafkaContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.kafka.template.kafkaContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.kafka.template.kafkaContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.kafka.template.kafkaContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.kafka.template.kafkaContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.kafka.template.kafkaContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.kafka.template.kafkaContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.kafka.template.kafkaContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.kafka.template.kafkaContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".

Property	Type	Description
spec.kafka.kafka.template.kafkaContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.kafka.template.kafkaContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.kafka.template.kafkaContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.kafka.template.kafkaContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.kafka.template.kafkaContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafka.template.kafkaContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.kafka.template.kafkaContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.kafka.template.kafkaContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.kafka.template.kafkaContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafka.template.kafkaContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.kafka.template.kafkaContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.kafka.template.kafkaContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".

Property	Type	Description
spec.kafka.kafka.template.kafkaContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.kafka.template.perPodIngress	object	Template for Kafka per-pod `Ingress` used for access from outside of Kubernetes.
spec.kafka.kafka.template.perPodIngress.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.perPodIngress.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafka.template.perPodIngress.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafka.template.perPodRoute	object	Template for Kafka per-pod `Routes` used for access from outside of OpenShift.
spec.kafka.kafka.template.perPodRoute.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.perPodRoute.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafka.template.perPodRoute.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafka.template.perPodService	object	Template for Kafka per-pod `Services` used for access from outside of Kubernetes.
spec.kafka.kafka.template.perPodService.externalTrafficPolicy	string	Specifies whether the service routes external traffic to node-local or cluster-wide endpoints. `Cluster` may cause a second hop to another node and obscures the client source IP. `Local` avoids a second hop for LoadBalancer and Nodeport type services and preserves the client source IP (when supported by the infrastructure). If unspecified, Kubernetes will use `Cluster` as the default.
spec.kafka.kafka.template.perPodService.loadBalancerSourceRanges	array	A list of CIDR ranges (for example `10.0.0.0/8` or `130.211.204.1/32`) from which clients can connect to load balancer type listeners. If supported by the platform, traffic through the loadbalancer is restricted to the specified CIDR ranges. This field is applicable only for loadbalancer type services and is ignored if the cloud provider does not support the feature. For more information, see https://v1-17.docs.kubernetes.io/docs/tasks/access-application-cluster/configure-cloud-provider-firewall/.
spec.kafka.kafka.template.perPodService.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.perPodService.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafka.template.perPodService.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafka.template.persistentVolumeClaim	object	Template for all Kafka `PersistentVolumeClaims`.
spec.kafka.kafka.template.persistentVolumeClaim.metadata	object	Metadata applied to the resource.

Property	Type	Description
spec.kafka.kafka.template.persistentVolumeClaim.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.persistentVolumeClaim.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.pod	object	Template for Kafka **`Pods`**.
spec.kafka.kafka.template.pod.affinity	object	The pod's affinity rules.
spec.kafka.kafka.template.pod.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.kafka.template.pod.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.kafka.template.pod.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.kafka.template.pod.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.template.pod.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.template.pod.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.kafka.template.pod.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".

Property	Type	Description
spec.kafka.kafka.template.pod.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafka.template.pod.hostAliases	array	The pod's HostAliases. HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified.
spec.kafka.kafka.template.pod.imagePullSecrets	array	List of references to secrets in the same namespace to use for pulling any of the images used by this Pod. When the **`STRIMZI_IMAGE_PULL_SECRETS`** environment variable in Cluster Operator and the **`imagePullSecrets`** option are specified, only the **`imagePullSecrets`** variable is used and the **`STRIMZI_IMAGE_PULL_SECRETS`** variable is ignored.
spec.kafka.kafka.template.pod.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.pod.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.pod.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafka.template.pod.priorityClassName	string	The name of the priority class used to assign priority to the pods. For more information about priority classes, see {K8sPriorityClass}.
spec.kafka.kafka.template.pod.schedulerName	string	The name of the scheduler used to dispatch this **`Pod`**. If not specified, the default scheduler will be used.
spec.kafka.kafka.template.pod.securityContext	object	Configures pod-level security attributes and common container settings.
spec.kafka.kafka.template.pod.securityContext.fsGroup	integer	FsGroup corresponds to the JSON schema field "fsGroup".
spec.kafka.kafka.template.pod.securityContext.fsGroupChangePolicy	string	FsGroupChangePolicy corresponds to the JSON schema field "fsGroupChangePolicy".
spec.kafka.kafka.template.pod.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.kafka.template.pod.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.kafka.template.pod.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.kafka.template.pod.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.kafka.template.pod.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.kafka.template.pod.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".

Property	Type	Description
spec.kafka.kafka.template.pod.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafka.template.pod.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.kafka.template.pod.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.kafka.template.pod.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.kafka.template.pod.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafka.template.pod.securityContext.supplementalGroups	array	SupplementalGroups corresponds to the JSON schema field "supplementalGroups".
spec.kafka.kafka.template.pod.securityContext.sysctls	array	Sysctls corresponds to the JSON schema field "sysctls".
spec.kafka.kafka.template.pod.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.kafka.template.pod.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.kafka.template.pod.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.kafka.template.pod.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.kafka.template.pod.terminationGracePeriodSeconds	integer	The grace period is the duration in seconds after the processes running in the pod are sent a termination signal, and the time when the processes are forcibly halted with a kill signal. Set this value to longer than the expected cleanup time for your process. Value must be a non-negative integer. A zero value indicates delete immediately. You might need to increase the grace period for very large Kafka clusters, so that the Kafka brokers have enough time to transfer their work to another broker before they are terminated. Defaults to 30 seconds.
spec.kafka.kafka.template.pod.tolerations	array	The pod's tolerations.
spec.kafka.kafka.template.pod.topologySpreadConstraints	array	The pod's topology spread constraints.
spec.kafka.kafka.template.podDisruptionBudget	object	Template for Kafka **`PodDisruptionBudget`**.

Property	Type	Description
spec.kafka.kafka.template.podDisruptionBudget.maxUnavailable	integer	Maximum number of unavailable pods to allow automatic Pod eviction. A Pod eviction is allowed when the **maxUnavailable** number of pods or fewer are unavailable after the eviction. Setting this value to 0 prevents all voluntary evictions, so the pods must be evicted manually. Defaults to 1.
spec.kafka.kafka.template.podDisruptionBudget.metadata	object	Metadata to apply to the **PodDistruptionBugetTemplate** resource.
spec.kafka.kafka.template.podDisruptionBudget.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **StatefulSets**, **Deployments**, **Pods**, and **Services**.
spec.kafka.kafka.template.podDisruptionBudget.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **StatefulSets**, **Deployments**, **Pods**, and **Services**.
spec.kafka.kafka.template.statefulset	object	Template for Kafka **StatefulSet**.
spec.kafka.kafka.template.statefulset.metadata	object	Metadata applied to the resource.
spec.kafka.kafka.template.statefulset.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **StatefulSets**, **Deployments**, **Pods**, and **Services**.
spec.kafka.kafka.template.statefulset.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **StatefulSets**, **Deployments**, **Pods**, and **Services**.
spec.kafka.kafka.template.statefulset.podManagementPolicy	string	PodManagementPolicy which will be used for this StatefulSet. Valid values are **Parallel** and **OrderedReady**. Defaults to **Parallel**.
spec.kafka.kafka.template.tlsSidecarContainer	object	Template for the Kafka broker TLS sidecar container.
spec.kafka.kafka.template.tlsSidecarContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.kafka.template.tlsSidecarContainer.securityContext	object	Security context for the container.
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".

Property	Type	Description
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".

Property	Type	Description
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.kafka.template.tlsSidecarContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.kafka.tlsSidecar	object	TLS sidecar configuration.
spec.kafka.kafka.tlsSidecar.image	string	The docker image for the container.
spec.kafka.kafka.tlsSidecar.livenessProbe	object	Pod liveness checking.
spec.kafka.kafka.tlsSidecar.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.kafka.tlsSidecar.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.kafka.tlsSidecar.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.kafka.tlsSidecar.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.kafka.tlsSidecar.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.kafka.tlsSidecar.logLevel	string	The log level for the TLS sidecar. Default value is `notice`.
spec.kafka.kafka.tlsSidecar.readinessProbe	object	Pod readiness checking.
spec.kafka.kafka.tlsSidecar.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.

Property	Type	Description
spec.kafka.kafka.tlsSidecar.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.kafka.tlsSidecar.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.kafka.tlsSidecar.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.kafka.tlsSidecar.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.kafka.tlsSidecar.resources	object	CPU and memory resources to reserve.
spec.kafka.kafka.tlsSidecar.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.kafka.tlsSidecar.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.kafka.tolerations	array	The pod's tolerations.
spec.kafka.kafka.version	string	The kafka broker version. Defaults to {DefaultKafkaVersion}. Consult the user documentation to understand the process required to upgrade or downgrade the version.
spec.kafka.kafkaExporter	object	Configuration of the Kafka Exporter. Kafka Exporter can provide additional metrics, for example lag of consumer group at topic/partition.
spec.kafka.kafkaExporter.enableSaramaLogging	boolean	Enable Sarama logging, a Go client library used by the Kafka Exporter.
spec.kafka.kafkaExporter.groupRegex	string	Regular expression to specify which consumer groups to collect. Default value is `.*`.
spec.kafka.kafkaExporter.image	string	The docker image for the pods.
spec.kafka.kafkaExporter.livenessProbe	object	Pod liveness check.
spec.kafka.kafkaExporter.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.kafkaExporter.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.kafkaExporter.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.kafkaExporter.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.kafkaExporter.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.kafkaExporter.logging	string	Only log messages with the given severity or above. Valid levels: [**`debug`**, **`info`**, **`warn`**, **`error`**, **`fatal`**]. Default log level is **`info`**.

Property	Type	Description
spec.kafka.kafkaExporter.readinessProbe	object	Pod readiness check.
spec.kafka.kafkaExporter.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.kafkaExporter.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.kafkaExporter.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.kafkaExporter.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.kafkaExporter.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.kafkaExporter.resources	object	CPU and memory resources to reserve.
spec.kafka.kafkaExporter.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.kafkaExporter.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.kafkaExporter.template	object	Customization of deployment templates and pods.
spec.kafka.kafkaExporter.template.container	object	Template for the Kafka Exporter container.
spec.kafka.kafkaExporter.template.container.env	array	Environment variables which should be applied to the container.
spec.kafka.kafkaExporter.template.container.securityContext	object	Security context for the container.
spec.kafka.kafkaExporter.template.container.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.kafkaExporter.template.container.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.kafkaExporter.template.container.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.kafkaExporter.template.container.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".

Property	Type	Description
spec.kafka.kafkaExporter.template.container.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.kafkaExporter.template.container.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.kafkaExporter.template.container.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.kafkaExporter.template.container.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.kafkaExporter.template.container.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.kafkaExporter.template.container.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.kafkaExporter.template.container.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.kafkaExporter.template.container.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.kafkaExporter.template.container.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.kafkaExporter.template.container.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafkaExporter.template.container.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.kafkaExporter.template.container.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.kafkaExporter.template.container.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".

Property	Type	Description
spec.kafka.kafkaExporter.template.container.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafkaExporter.template.container.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.kafkaExporter.template.container.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.kafkaExporter.template.container.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.kafkaExporter.template.container.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.kafkaExporter.template.deployment	object	Template for Kafka Exporter **`Deployment`**.
spec.kafka.kafkaExporter.template.deployment.metadata	object	Metadata applied to the resource.
spec.kafka.kafkaExporter.template.deployment.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafkaExporter.template.deployment.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.kafkaExporter.template.pod	object	Template for Kafka Exporter **`Pods`**.
spec.kafka.kafkaExporter.template.pod.affinity	object	The pod's affinity rules.
spec.kafka.kafkaExporter.template.pod.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.kafkaExporter.template.pod.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".

Property	Type	Description
spec.kafka.kafkaExporter.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafkaExporter.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.kafkaExporter.template.pod.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.kafkaExporter.template.pod.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafkaExporter.template.pod.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafkaExporter.template.pod.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.kafkaExporter.template.pod.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafkaExporter.template.pod.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.kafkaExporter.template.pod.hostAliases	array	The pod's HostAliases. HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified.
spec.kafka.kafkaExporter.template.pod.imagePullSecrets	array	List of references to secrets in the same namespace to use for pulling any of the images used by this Pod. When the **`STRIMZI_IMAGE_PULL_SECRETS`** environment variable in Cluster Operator and the **`imagePullSecrets`** option are specified, only the **`imagePullSecrets`** variable is used and the **`STRIMZI_IMAGE_PULL_SECRETS`** variable is ignored.
spec.kafka.kafkaExporter.template.pod.metadata	object	Metadata applied to the resource.
spec.kafka.kafkaExporter.template.pod.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.

Property	Type	Description
spec.kafka.kafkaExporter.template.pod.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafkaExporter.template.pod.priorityClassName	string	The name of the priority class used to assign priority to the pods. For more information about priority classes, see {K8sPriorityClass}.
spec.kafka.kafkaExporter.template.pod.schedulerName	string	The name of the scheduler used to dispatch this `Pod`. If not specified, the default scheduler will be used.
spec.kafka.kafkaExporter.template.pod.securityContext	object	Configures pod-level security attributes and common container settings.
spec.kafka.kafkaExporter.template.pod.securityContext.fsGroup	integer	FsGroup corresponds to the JSON schema field "fsGroup".
spec.kafka.kafkaExporter.template.pod.securityContext.fsGroupChangePolicy	string	FsGroupChangePolicy corresponds to the JSON schema field "fsGroupChangePolicy".
spec.kafka.kafkaExporter.template.pod.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.kafkaExporter.template.pod.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.kafkaExporter.template.pod.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.kafkaExporter.template.pod.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.kafkaExporter.template.pod.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.kafkaExporter.template.pod.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.kafkaExporter.template.pod.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafkaExporter.template.pod.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.kafkaExporter.template.pod.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".

Property	Type	Description
spec.kafka.kafkaExporter.template.pod.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.kafkaExporter.template.pod.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.kafkaExporter.template.pod.securityContext.supplementalGroups	array	SupplementalGroups corresponds to the JSON schema field "supplementalGroups".
spec.kafka.kafkaExporter.template.pod.securityContext.sysctls	array	Sysctls corresponds to the JSON schema field "sysctls".
spec.kafka.kafkaExporter.template.pod.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.kafkaExporter.template.pod.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.kafkaExporter.template.pod.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.kafkaExporter.template.pod.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.kafkaExporter.template.pod.terminationGracePeriodSeconds	integer	The grace period is the duration in seconds after the processes running in the pod are sent a termination signal, and the time when the processes are forcibly halted with a kill signal. Set this value to longer than the expected cleanup time for your process. Value must be a non-negative integer. A zero value indicates delete immediately. You might need to increase the grace period for very large Kafka clusters, so that the Kafka brokers have enough time to transfer their work to another broker before they are terminated. Defaults to 30 seconds.
spec.kafka.kafkaExporter.template.pod.tolerations	array	The pod's tolerations.
spec.kafka.kafkaExporter.template.pod.topologySpreadConstraints	array	The pod's topology spread constraints.
spec.kafka.kafkaExporter.template.service	object	Template for Kafka Exporter **`Service`**.
spec.kafka.kafkaExporter.template.service.metadata	object	Metadata applied to the resource.
spec.kafka.kafkaExporter.template.service.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.

Property	Type	Description
spec.kafka.kafkaExporter.template.service.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as `StatefulSets`, `Deployments`, `Pods`, and `Services`.
spec.kafka.kafkaExporter.topicRegex	string	Regular expression to specify which topics to collect. Default value is `.*`.
spec.kafka.maintenanceTimeWindows	array	A list of time windows for maintenance tasks (that is, certificates renewal). Each time window is defined by a cron expression.
spec.kafka.topicOperator	object	Configuration of the Topic Operator.
spec.kafka.topicOperator.affinity	object	Pod affinity rules.
spec.kafka.topicOperator.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.topicOperator.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.topicOperator.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.topicOperator.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.topicOperator.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.topicOperator.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.topicOperator.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.topicOperator.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.topicOperator.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".

Property	Type	Description
spec.kafka.topicOperator.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.topicOperator.image	string	The image to use for the Topic Operator.
spec.kafka.topicOperator.jvmOptions	object	JVM Options for pods.
spec.kafka.topicOperator.jvmOptions.-XX	object	A map of -XX options to the JVM.
spec.kafka.topicOperator.jvmOptions.-Xms	string	-Xms option to to the JVM.
spec.kafka.topicOperator.jvmOptions.-Xmx	string	-Xmx option to to the JVM.
spec.kafka.topicOperator.jvmOptions.gcLoggingEnabled	boolean	Specifies whether the Garbage Collection logging is enabled. The default is false.
spec.kafka.topicOperator.jvmOptions.javaSystemProperties	array	A map of additional system properties which will be passed using the `-D` option to the JVM.
spec.kafka.topicOperator.livenessProbe	object	Pod liveness checking.
spec.kafka.topicOperator.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.topicOperator.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.topicOperator.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.topicOperator.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.topicOperator.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.topicOperator.logging	object	Logging configuration.
spec.kafka.topicOperator.logging.loggers	object	A Map from logger name to logger level.
spec.kafka.topicOperator.logging.name	string	The name of the `ConfigMap` from which to get the logging configuration.
spec.kafka.topicOperator.logging.type	string	Logging type, must be either 'inline' or 'external'.

Property	Type	Description
spec.kafka.topicOperator.readinessProbe	object	Pod readiness checking.
spec.kafka.topicOperator.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.topicOperator.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.topicOperator.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.topicOperator.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.topicOperator.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.topicOperator.reconciliationIntervalSeconds	integer	Interval between periodic reconciliations.
spec.kafka.topicOperator.resources	object	CPU and memory resources to reserve.
spec.kafka.topicOperator.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.topicOperator.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.topicOperator.tlsSidecar	object	TLS sidecar configuration.
spec.kafka.topicOperator.tlsSidecar.image	string	The docker image for the container.
spec.kafka.topicOperator.tlsSidecar.livenessProbe	object	Pod liveness checking.
spec.kafka.topicOperator.tlsSidecar.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.topicOperator.tlsSidecar.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.topicOperator.tlsSidecar.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.topicOperator.tlsSidecar.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.topicOperator.tlsSidecar.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.

Property	Type	Description
spec.kafka.topicOperator.tlsSidecar.logLevel	string	The log level for the TLS sidecar. Default value is `notice`.
spec.kafka.topicOperator.tlsSidecar.readinessProbe	object	Pod readiness checking.
spec.kafka.topicOperator.tlsSidecar.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.topicOperator.tlsSidecar.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.topicOperator.tlsSidecar.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.topicOperator.tlsSidecar.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.topicOperator.tlsSidecar.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.topicOperator.tlsSidecar.resources	object	CPU and memory resources to reserve.
spec.kafka.topicOperator.tlsSidecar.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.topicOperator.tlsSidecar.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.topicOperator.topicMetadataMaxAttempts	integer	The number of attempts at getting topic metadata.
spec.kafka.topicOperator.watchedNamespace	string	The namespace the Topic Operator should watch.
spec.kafka.topicOperator.zookeeperSessionTimeoutSeconds	integer	Timeout for the ZooKeeper session.
spec.kafka.zookeeper	object	Configuration of the ZooKeeper cluster.
spec.kafka.zookeeper.affinity	object	The pod's affinity rules.
spec.kafka.zookeeper.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.zookeeper.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".

Property	Type	Description
spec.kafka.zookeeper.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".
spec.kafka.zookeeper.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.zookeeper.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.zookeeper.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.config	object	The ZooKeeper broker config. Properties with the following prefixes cannot be set: server., dataDir, dataLogDir, clientPort, authProvider, quorum.auth, requireClientAuthScheme, snapshot.trust.empty, standaloneEnabled, reconfigEnabled, 4lw.commands.whitelist, secureClientPort, ssl., serverCnxnFactory, sslQuorum (with the exception of: ssl.protocol, ssl.quorum.protocol, ssl.enabledProtocols, ssl.quorum.enabledProtocols, ssl.ciphersuites, ssl.quorum.ciphersuites, ssl.hostnameVerification, ssl.quorum.hostnameVerification).
spec.kafka.zookeeper.image	string	The docker image for the pods.
spec.kafka.zookeeper.jvmOptions	object	JVM Options for pods.
spec.kafka.zookeeper.jvmOptions.-XX	object	A map of -XX options to the JVM.
spec.kafka.zookeeper.jvmOptions.-Xms	string	-Xms option to to the JVM.
spec.kafka.zookeeper.jvmOptions.-Xmx	string	-Xmx option to to the JVM.
spec.kafka.zookeeper.jvmOptions.gcLoggingEnabled	boolean	Specifies whether the Garbage Collection logging is enabled. The default is false.

Property	Type	Description
spec.kafka.zookeeper.jvmOptions.javaSystemProperties	array	A map of additional system properties which will be passed using the `-D` option to the JVM.
spec.kafka.zookeeper.livenessProbe	object	Pod liveness checking.
spec.kafka.zookeeper.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.zookeeper.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.zookeeper.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.zookeeper.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.zookeeper.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.zookeeper.logging	object	Logging configuration for ZooKeeper.
spec.kafka.zookeeper.logging.loggers	object	A Map from logger name to logger level.
spec.kafka.zookeeper.logging.name	string	The name of the **`ConfigMap`** from which to get the logging configuration.
spec.kafka.zookeeper.logging.type	string	Logging type, must be either 'inline' or 'external'.
spec.kafka.zookeeper.metrics	object	The Prometheus JMX Exporter configuration. See https://github.com/prometheus/jmx_exporter for details of the structure of this configuration.
spec.kafka.zookeeper.metricsConfig	object	Metrics configuration.
spec.kafka.zookeeper.metricsConfig.type	string	Metrics type. Only 'jmxPrometheusExporter' supported currently.
spec.kafka.zookeeper.metricsConfig.valueFrom	object	ConfigMap where the Prometheus JMX Exporter configuration is stored. For details of the structure of this configuration, see the {JMXExporter}.
spec.kafka.zookeeper.metricsConfig.valueFrom.configMapKeyRef	object	Reference to the key in the ConfigMap containing the metrics configuration.
spec.kafka.zookeeper.metricsConfig.valueFrom.configMapKeyRef.key	string	Key corresponds to the JSON schema field "key".
spec.kafka.zookeeper.metricsConfig.valueFrom.configMapKeyRef.name	string	Name corresponds to the JSON schema field "name".
spec.kafka.zookeeper.metricsConfig.valueFrom.configMapKeyRef.optional	boolean	Optional corresponds to the JSON schema field "optional".
spec.kafka.zookeeper.readinessProbe	object	Pod readiness checking.

Property	Type	Description
spec.kafka.zookeeper.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.zookeeper.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.zookeeper.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.zookeeper.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.zookeeper.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.zookeeper.replicas	integer	The number of pods in the cluster.
spec.kafka.zookeeper.resources	object	CPU and memory resources to reserve.
spec.kafka.zookeeper.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.zookeeper.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.zookeeper.storage	object	Storage configuration (disk). Cannot be updated.
spec.kafka.zookeeper.storage.class	string	The storage class to use for dynamic volume allocation.
spec.kafka.zookeeper.storage.deleteClaim	boolean	Specifies if the persistent volume claim has to be deleted when the cluster is un-deployed.
spec.kafka.zookeeper.storage.id	integer	Storage identification number. It is mandatory only for storage volumes defined in a storage of type 'jbod'.
spec.kafka.zookeeper.storage.overrides	array	Overrides for individual brokers. The **`overrides`** field allows to specify a different configuration for different brokers.
spec.kafka.zookeeper.storage.selector	object	Specifies a specific persistent volume to use. It contains key:value pairs representing labels for selecting such a volume.
spec.kafka.zookeeper.storage.size	string	When type=persistent-claim, defines the size of the persistent volume claim (i.e 1Gi). Mandatory when type=persistent-claim.
spec.kafka.zookeeper.storage.sizeLimit	string	When type=ephemeral, defines the total amount of local storage required for this EmptyDir volume (for example 1Gi).
spec.kafka.zookeeper.storage.type	string	Storage type, must be either 'ephemeral' or 'persistent-claim'.
spec.kafka.zookeeper.template	object	Template for ZooKeeper cluster resources. The template allows users to specify how are the **`StatefulSet`**, **`Pods`** and **`Services`** generated.
spec.kafka.zookeeper.template.clientService	object	Template for ZooKeeper client **`Service`**.
spec.kafka.zookeeper.template.clientService.metadata	object	Metadata applied to the resource.
spec.kafka.zookeeper.template.clientService.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.

Property	Type	Description
spec.kafka.zookeeper.template.clientService.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.nodesService	object	Template for ZooKeeper nodes **`Service`**.
spec.kafka.zookeeper.template.nodesService.metadata	object	Metadata applied to the resource.
spec.kafka.zookeeper.template.nodesService.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.nodesService.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.persistentVolumeClaim	object	Template for all ZooKeeper **`PersistentVolumeClaims`**.
spec.kafka.zookeeper.template.persistentVolumeClaim.metadata	object	Metadata applied to the resource.
spec.kafka.zookeeper.template.persistentVolumeClaim.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.persistentVolumeClaim.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.pod	object	Template for ZooKeeper **`Pods`**.
spec.kafka.zookeeper.template.pod.affinity	object	The pod's affinity rules.
spec.kafka.zookeeper.template.pod.affinity.nodeAffinity	object	NodeAffinity corresponds to the JSON schema field "nodeAffinity".
spec.kafka.zookeeper.template.pod.affinity.nodeAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution	object	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.template.pod.affinity.nodeAffinity.requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms	array	NodeSelectorTerms corresponds to the JSON schema field "nodeSelectorTerms".

Property	Type	Description
spec.kafka.zookeeper.template.pod.affinity.podAffinity	object	PodAffinity corresponds to the JSON schema field "podAffinity".
spec.kafka.zookeeper.template.pod.affinity.podAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.template.pod.affinity.podAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.template.pod.affinity.podAntiAffinity	object	PodAntiAffinity corresponds to the JSON schema field "podAntiAffinity".
spec.kafka.zookeeper.template.pod.affinity.podAntiAffinity.preferredDuringSchedulingIgnoredDuringExecution	array	PreferredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "preferredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.template.pod.affinity.podAntiAffinity.requiredDuringSchedulingIgnoredDuringExecution	array	RequiredDuringSchedulingIgnoredDuringExecution corresponds to the JSON schema field "requiredDuringSchedulingIgnoredDuringExecution".
spec.kafka.zookeeper.template.pod.hostAliases	array	The pod's HostAliases. HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified.
spec.kafka.zookeeper.template.pod.imagePullSecrets	array	List of references to secrets in the same namespace to use for pulling any of the images used by this Pod. When the **`STRIMZI_IMAGE_PULL_SECRETS`** environment variable in Cluster Operator and the **`imagePullSecrets`** option are specified, only the **`imagePullSecrets`** variable is used and the **`STRIMZI_IMAGE_PULL_SECRETS`** variable is ignored.
spec.kafka.zookeeper.template.pod.metadata	object	Metadata applied to the resource.
spec.kafka.zookeeper.template.pod.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.pod.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.pod.priorityClassName	string	The name of the priority class used to assign priority to the pods. For more information about priority classes, see {K8sPriorityClass}.
spec.kafka.zookeeper.template.pod.schedulerName	string	The name of the scheduler used to dispatch this **`Pod`**. If not specified, the default scheduler will be used.
spec.kafka.zookeeper.template.pod.securityContext	object	Configures pod-level security attributes and common container settings.
spec.kafka.zookeeper.template.pod.securityContext.fsGroup	integer	FsGroup corresponds to the JSON schema field "fsGroup".

Property	Type	Description
spec.kafka.zookeeper.template.pod.securityContext.fsGroupChangePolicy	string	FsGroupChangePolicy corresponds to the JSON schema field "fsGroupChangePolicy".
spec.kafka.zookeeper.template.pod.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.zookeeper.template.pod.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".
spec.kafka.zookeeper.template.pod.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.zookeeper.template.pod.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.zookeeper.template.pod.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.zookeeper.template.pod.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.zookeeper.template.pod.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.zookeeper.template.pod.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.zookeeper.template.pod.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.zookeeper.template.pod.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.zookeeper.template.pod.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.zookeeper.template.pod.securityContext.supplementalGroups	array	SupplementalGroups corresponds to the JSON schema field "supplementalGroups".
spec.kafka.zookeeper.template.pod.securityContext.sysctls	array	Sysctls corresponds to the JSON schema field "sysctls".
spec.kafka.zookeeper.template.pod.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".

Property	Type	Description
spec.kafka.zookeeper.template.pod.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.zookeeper.template.pod.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".
spec.kafka.zookeeper.template.pod.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.zookeeper.template.pod.terminationGracePeriodSeconds	integer	The grace period is the duration in seconds after the processes running in the pod are sent a termination signal, and the time when the processes are forcibly halted with a kill signal. Set this value to longer than the expected cleanup time for your process. Value must be a non-negative integer. A zero value indicates delete immediately. You might need to increase the grace period for very large Kafka clusters, so that the Kafka brokers have enough time to transfer their work to another broker before they are terminated. Defaults to 30 seconds.
spec.kafka.zookeeper.template.pod.tolerations	array	The pod's tolerations.
spec.kafka.zookeeper.template.pod.topologySpreadConstraints	array	The pod's topology spread constraints.
spec.kafka.zookeeper.template.podDisruptionBudget	object	Template for ZooKeeper **`PodDisruptionBudget`**.
spec.kafka.zookeeper.template.podDisruptionBudget.maxUnavailable	integer	Maximum number of unavailable pods to allow automatic Pod eviction. A Pod eviction is allowed when the **`maxUnavailable`** number of pods or fewer are unavailable after the eviction. Setting this value to 0 prevents all voluntary evictions, so the pods must be evicted manually. Defaults to 1.
spec.kafka.zookeeper.template.podDisruptionBudget.metadata	object	Metadata to apply to the **`PodDistruptionBugetTemplate`** resource.
spec.kafka.zookeeper.template.podDisruptionBudget.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.podDisruptionBudget.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.statefulset	object	Template for ZooKeeper **`StatefulSet`**.
spec.kafka.zookeeper.template.statefulset.metadata	object	Metadata applied to the resource.
spec.kafka.zookeeper.template.statefulset.metadata.annotations	object	Annotations added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.
spec.kafka.zookeeper.template.statefulset.metadata.labels	object	Labels added to the resource template. Can be applied to different resources such as **`StatefulSets`**, **`Deployments`**, **`Pods`**, and **`Services`**.

Property	Type	Description
spec.kafka.zookeeper.template.statefulset.podManagementPolicy	string	PodManagementPolicy which will be used for this StatefulSet. Valid values are `Parallel` and `OrderedReady`. Defaults to `Parallel`.
spec.kafka.zookeeper.template.tlsSidecarContainer	object	Template for the Zookeeper server TLS sidecar container. The TLS sidecar is not used anymore and this option will be ignored.
spec.kafka.zookeeper.template.tlsSidecarContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext	object	Security context for the container.
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".

Property	Type	Description
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".

Property	Type	Description
spec.kafka.zookeeper.template.tlsSidecarContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.zookeeper.template.zookeeperContainer	object	Template for the ZooKeeper container.
spec.kafka.zookeeper.template.zookeeperContainer.env	array	Environment variables which should be applied to the container.
spec.kafka.zookeeper.template.zookeeperContainer.securityContext	object	Security context for the container.
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.allowPrivilegeEscalation	boolean	AllowPrivilegeEscalation corresponds to the JSON schema field "allowPrivilegeEscalation".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.capabilities	object	Capabilities corresponds to the JSON schema field "capabilities".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.capabilities.add	array	Add corresponds to the JSON schema field "add".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.capabilities.drop	array	Drop corresponds to the JSON schema field "drop".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.privileged	boolean	Privileged corresponds to the JSON schema field "privileged".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.procMount	string	ProcMount corresponds to the JSON schema field "procMount".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.readOnlyRootFilesystem	boolean	ReadOnlyRootFilesystem corresponds to the JSON schema field "readOnlyRootFilesystem".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.runAsGroup	integer	RunAsGroup corresponds to the JSON schema field "runAsGroup".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.runAsNonRoot	boolean	RunAsNonRoot corresponds to the JSON schema field "runAsNonRoot".

Property	Type	Description
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.runAsUser	integer	RunAsUser corresponds to the JSON schema field "runAsUser".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.seLinuxOptions	object	SeLinuxOptions corresponds to the JSON schema field "seLinuxOptions".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.seLinuxOptions.level	string	Level corresponds to the JSON schema field "level".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.seLinuxOptions.role	string	Role corresponds to the JSON schema field "role".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.seLinuxOptions.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.seLinuxOptions.user	string	User corresponds to the JSON schema field "user".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.seccompProfile	object	SeccompProfile corresponds to the JSON schema field "seccompProfile".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.seccompProfile.localhostProfile	string	LocalhostProfile corresponds to the JSON schema field "localhostProfile".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.seccompProfile.type	string	Type corresponds to the JSON schema field "type".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.windowsOptions	object	WindowsOptions corresponds to the JSON schema field "windowsOptions".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.windowsOptions.gmsaCredentialSpec	string	GmsaCredentialSpec corresponds to the JSON schema field "gmsaCredentialSpec".
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.windowsOptions.gmsaCredentialSpecName	string	GmsaCredentialSpecName corresponds to the JSON schema field "gmsaCredentialSpecName".

Property	Type	Description
spec.kafka.zookeeper.template.zookeeperContainer.securityContext.windowsOptions.runAsUserName	string	RunAsUserName corresponds to the JSON schema field "runAsUserName".
spec.kafka.zookeeper.tlsSidecar	object	TLS sidecar configuration. The TLS sidecar is not used anymore and this option will be ignored.
spec.kafka.zookeeper.tlsSidecar.image	string	The docker image for the container.
spec.kafka.zookeeper.tlsSidecar.livenessProbe	object	Pod liveness checking.
spec.kafka.zookeeper.tlsSidecar.livenessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.zookeeper.tlsSidecar.livenessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.zookeeper.tlsSidecar.livenessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.zookeeper.tlsSidecar.livenessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.zookeeper.tlsSidecar.livenessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.zookeeper.tlsSidecar.logLevel	string	The log level for the TLS sidecar. Default value is **`notice`**.
spec.kafka.zookeeper.tlsSidecar.readinessProbe	object	Pod readiness checking.
spec.kafka.zookeeper.tlsSidecar.readinessProbe.failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
spec.kafka.zookeeper.tlsSidecar.readinessProbe.initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
spec.kafka.zookeeper.tlsSidecar.readinessProbe.periodSeconds	integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
spec.kafka.zookeeper.tlsSidecar.readinessProbe.successThreshold	integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
spec.kafka.zookeeper.tlsSidecar.readinessProbe.timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
spec.kafka.zookeeper.tlsSidecar.resources	object	CPU and memory resources to reserve.

Property	Type	Description
spec.kafka.zookeeper.tlsSidecar.resources.limits	object	Limits corresponds to the JSON schema field "limits".
spec.kafka.zookeeper.tlsSidecar.resources.requests	object	Requests corresponds to the JSON schema field "requests".
spec.kafka.zookeeper.tolerations	array	The pod's tolerations.
spec.kafkaAccessSecret	string	KafkaAccessSecret when set will configure the application to use the provided kafka credentials instead of deploying its own.
spec.license	object	LicenseAccept is used to accept the license agreement
spec.license.accept	boolean	(No Description)
spec.size	string	(No Description)
spec.storageClass	string	StorageClass is the storage class used for PVCs. If not set it will fall back to the cluster default when available.

status

Description: CommonStatus that can be used with StatusUpdater

Type: object

Table 1. Description of AIOpsEdge/v1beta1 status

Property	Type	Description
status	object	CommonStatus that can be used with StatusUpdater
status.conditions	array	(No Description)
status.observedGeneration	integer	(No Description)
status.phase	string	(No Description)
status.resources	object	Resources contained in status

AIOpsUI/v1

For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms. AIOpsUI is the Schema for the aiopsuis API.

spec

Description: (No Description)

Type: object

Table 1. Description of AIOpsUI/v1 spec

Property	Type	Description
spec	object	(No Description)
spec.configuration	object	Additional configuration for the UI deployment
spec.configuration.topologyModel	string	TopologyModel - Specify the terminology to use in IBM Cloud Pak for AIOps when defining collections of topology resource groups. Valid values are "application" or "service". If this setting is not specified, the default value will be "application".
spec.container	object	ContainerSpec - possible configurations around containers
spec.container.applicationUiApi	object	(No Description)
spec.container.applicationUiApi.container	object	ContainerSpec - possible configurations around containers
spec.container.applicationUiApi.container.containerPort	integer	(No Description)
spec.container.applicationUiApi.container.env	array	(No Description)

Property	Type	Description
spec.container.applicationUiApi.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.applicationUiApi.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.applicationUiApi.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.applicationUiApi.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.applicationUiApi.image	object	ImageSpec - specification around how to pull an image.
spec.container.applicationUiApi.image.path	string	(No Description)
spec.container.applicationUiApi.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.container.applicationUiApi.image.pullSecret	string	(No Description)
spec.container.applicationUiApi.tlsSecretName	string	(No Description)
spec.container.containerPort	integer	(No Description)
spec.container.env	array	(No Description)
spec.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiApi	object	(No Description)
spec.container.uiApi.container	object	ContainerSpec - possible configurations around containers
spec.container.uiApi.container.containerPort	integer	(No Description)
spec.container.uiApi.container.env	array	(No Description)
spec.container.uiApi.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.uiApi.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.uiApi.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Property	Type	Description
spec.container.uiApi.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiApi.image	object	ImageSpec - specification around how to pull an image.
spec.container.uiApi.image.path	string	(No Description)
spec.container.uiApi.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.container.uiApi.image.pullSecret	string	(No Description)
spec.container.uiApi.tlsSecretName	string	(No Description)
spec.container.uiBundleApi	object	(No Description)
spec.container.uiBundleApi.container	object	ContainerSpec - possible configurations around containers
spec.container.uiBundleApi.container.containerPort	integer	(No Description)
spec.container.uiBundleApi.container.env	array	(No Description)
spec.container.uiBundleApi.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.uiBundleApi.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.uiBundleApi.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiBundleApi.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiBundleApi.image	object	ImageSpec - specification around how to pull an image.
spec.container.uiBundleApi.image.path	string	(No Description)
spec.container.uiBundleApi.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.container.uiBundleApi.image.pullSecret	string	(No Description)
spec.container.uiBundleApi.tlsSecretName	string	(No Description)
spec.container.uiContent	object	(No Description)
spec.container.uiContent.container	object	ContainerSpec - possible configurations around containers
spec.container.uiContent.container.containerPort	integer	(No Description)
spec.container.uiContent.container.env	array	(No Description)
spec.container.uiContent.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.uiContent.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		

Property	Type	Description
spec.container.uiContent.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiContent.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiContent.image	object	ImageSpec - specification around how to pull an image.
spec.container.uiContent.image.path	string	(No Description)
spec.container.uiContent.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.container.uiContent.image.pullSecret	string	(No Description)
spec.container.uiContent.tlsSecretName	string	(No Description)
spec.container.uiServer	object	(No Description)
spec.container.uiServer.container	object	ContainerSpec - possible configurations around containers
spec.container.uiServer.container.containerPort	integer	(No Description)
spec.container.uiServer.container.env	array	(No Description)
spec.container.uiServer.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.uiServer.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.uiServer.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiServer.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiServer.image	object	ImageSpec - specification around how to pull an image.
spec.container.uiServer.image.path	string	(No Description)
spec.container.uiServer.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.container.uiServer.image.pullSecret	string	(No Description)
spec.container.uiServer.tlsSecretName	string	(No Description)
spec.image	object	ImageSpec - specification around how to pull an image.
spec.image.path	string	(No Description)
spec.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.image.pullSecret	string	(No Description)
spec.license	object	License - license spec determines whether a license has been accepted
spec.license.accept	boolean	License acceptance switch
spec.name	string	Name of the service being deployed
spec.redis	object	Name of the service being deployed
spec.redis.bindingSecret	string	Data store binding secret
spec.replicas	integer	(No Description)
spec.size	string	(No Description)
spec.tlsSecretName	string	(No Description)
spec.version	string	Version of the csv deploying
spec.zenserviceName	string	name of the zenservice cr that is used for this deployment

status

Description: (No Description)

Type: object

Table 1. Description of AIOpsUI/v1 status

Property	Type	Description
status	object	(No Description)
status.conditions	array	(No Description)
status.managedResources	array	(No Description)
status.phase	string	(No Description)
status.versions	object	(No Description)
status.versions.reconciled	string	(No Description)

AIOpsUIExtension/v1

AIOpsUIExtension is the Schema for the aiopsuiextensions API

spec

Description: AIOpsUIExtensionSpec defines the desired state of AIOpsUIExtension

Type: object

Table 1. Description of AIOpsUIExtension/v1 spec

Property	Type	Description
spec	object	AIOpsUIExtensionSpec defines the desired state of AIOpsUIExtension
spec.menuRoutes	(No Type)	MenuRoutes contains menu configurations, that will be mapped to the formats required by Zen and Akora.
spec.routes	(No Type)	Routes contains page configuration and bundle configurations.

status

Description: (No Description)

Type: object

Table 1. Description of AIOpsUIExtension/v1 status

Property	Type	Description
status	object	(No Description)
status.conditions	array	(No Description)
status.managedResources	array	(No Description)
status.phase	string	(No Description)
status.versions	object	(No Description)
status.versions.reconciled	string	(No Description)

AIManager/v1beta1

AIManager Deploys the IBM AIOps AI Manager service components of IBM Cloud Pak for AIOps. For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.

spec

Description: AIManagerSpec defines the desired state of AIManager

Type: object

Table 1. Description of AIManager/v1beta1 spec

Property	Type	Description
spec	object	AIManagerSpec defines the desired state of AIManager
spec.aimanagerValues	object	Plumbing Values Override (hidden)
spec.cartridge	object	Cartridge Name
spec.cartridge.name	string	Cartridge Name
spec.chatopsAuthentication	object	ChatopsAuthentication (hidden)
spec.chatopsAuthentication.configmapName	string	(No Description)
spec.chatopsAuthentication.secretName	string	(No Description)
spec.debug	object	Debug Options (hidden)
spec.debug.devPullSecrets	array	(No Description)
spec.debug.disableAIManager	boolean	(No Description)
spec.debug.disableElastic	boolean	(No Description)
spec.debug.disableKafka	boolean	(No Description)
spec.debug.disableModeltrain	boolean	Note: this element is no longer used and should be removed in the next major version of this API
spec.debug.disableZen	boolean	(No Description)
spec.debug.enableCustomSCC	boolean	(No Description)
spec.debug.useDockerIOForKafka	boolean	(No Description)
spec.directFlink	boolean	(No Description)
spec.elasticsearch	object	Elastic Search
spec.elasticsearch.elasticSuperuserSecretName	string	ElasticUserSecretName
spec.elasticsearch.elasticUserSecretName	string	ElasticUserSecretName
spec.eventProcessor	object	Event Processor
spec.eventProcessor.name	string	Event Processor Name
spec.existingPullSecret	object	Existing Pull Secret
spec.existingPullSecret.name	string	Secret Name
spec.flink	object	Flink
spec.flink.secretName	string	Flink Secret Name
spec.integratorProxy	object	Integrator Proxy Config Properties
spec.integratorProxy.enabled	boolean	A proxy outgoing Chatops routes through.
spec.integratorProxy.proxyRoute	string	(No Description)
spec.kafka	object	Kafka
spec.kafka.secretName	string	Kafka Secret Name
spec.license	object	Please accept the licenses to allow for installation
spec.license.accept	boolean	I have read and accept the license: https://ibm.biz/cp4aiops-421-license
spec.logAnomaly	object	LogAnomalyDetectorConfig (hidden)
spec.logAnomaly.limits	object	(No Description)
spec.logAnomaly.limits.cpu	string	(No Description)
spec.logAnomaly.limits.memory	string	(No Description)
spec.logAnomaly.replicas	integer	(No Description)
spec.logAnomaly.requests	object	(No Description)
spec.logAnomaly.requests.cpu	string	(No Description)
spec.logAnomaly.requests.memory	string	(No Description)
spec.logAnomaly.serverPools	integer	(No Description)
spec.minio	object	Minio Properties
spec.minio.customSizing	object	MinioCustomResources (hidden)
spec.minio.customSizing.limits	object	(No Description)
spec.minio.customSizing.limits.cpu	string	(No Description)

Property	Type	Description
spec.minio.customSizing.limits.memory	string	(No Description)
spec.minio.customSizing.replicas	integer	(No Description)
spec.minio.customSizing.requests	object	(No Description)
spec.minio.customSizing.requests.cpu	string	(No Description)
spec.minio.customSizing.requests.memory	string	(No Description)
spec.minio.customSizing.serverPools	integer	(No Description)
spec.minio.storage	object	The Minio storage options
spec.minio.storage.storageClassName	string	The StorageClassName
spec.minio.storage.storageSize	string	The Size of Storage (format example "100Gi")
spec.modelTrain	object	Model Train Properties
spec.modelTrain.maxLearners	integer	Maximum number of training to run at once
spec.modelTrain.pollInterval	integer	Poll Interval for the Queue
spec.modelTrain.sizeLimit	integer	Size Limit for the Queue
spec.postgresDB	object	Postgres Database Instance Properties
spec.postgresDB.edbSecretName	string	EDB Postgresql Database Connection Secret Name
spec.postgresDB.secretName	string	Postgresql Database Connection Secret Name
spec.redis	object	Redis
spec.redis.bindingSecretName	string	Redis Binding Secret Name
spec.repo	string	The private registry to pull images from instead of cp.icr.io. Leave blank if a private registry is not used. All images must be pushed to repo via mirroring.
spec.size	string	Preconfigured HA Sizing Options
spec.storage	object	The storage options
spec.storage.enabled	boolean	Enable Storage
spec.storage.storageClassName	string	The StorageClassName
spec.version	string	Version
spec.zen	object	The Zen options
spec.zen.deploy	boolean	Flag to deploy our own cpd in the same namespace as aimanager
spec.zen.ignoreReady	boolean	Flag to deploy zen customization even if not in ready state
spec.zen.instanceId	string	ID of Zen Service Instance
spec.zen.serviceInstanceName	string	Name of the ZenService Instance object
spec.zen.serviceNamespace	string	Namespace of the ZenService Instance
spec.zen.storage	object	The Zen storage options
spec.zen.storage.storageClassName	string	The StorageClassName

status

Description: AIManagerStatus defines the observed state of AIManager

Type: object

Table 1. Description of AIManager/v1beta1 status

Property	Type	Description
status	object	AIManagerStatus defines the observed state of AIManager
status.message	string	Status information from AIManager
status.phase	string	Current phase of AIManager.
status.versions	object	Information on versions for AI Manager
status.versions.available	object	Available properties
status.versions.available.versions	array	Possible available versions for AI Manager
status.versions.reconciled	string	Current version of the reconciled instance

Algorithm/v1alpha1

Algorithm is the Schema for the algorithms API

spec

Description: AlgorithmSpec defines the desired state of Algorithm

Type: object

Table 1. Description of Algorithm/v1alpha1 spec

Property	Type	Description
spec	object	AlgorithmSpec defines the desired state of Algorithm
spec.description	string	(No Description)
spec.enabled	boolean	(No Description)
spec.manifestBase64	string	(No Description)
spec.name	string	(No Description)
spec.registered	boolean	(No Description)
spec.runtime	string	(No Description)
spec.usedStatus	array	(No Description)
spec.version	string	(No Description)

status

Description: AlgorithmStatus defines the observed state of Algorithm

Type: object

Table 1. Description of Algorithm/v1alpha1 status

Property	Type	Description
status	object	AlgorithmStatus defines the observed state of Algorithm

Algorithm/v1beta1

Algorithm is the Schema for the algorithms API

spec

Description: AlgorithmSpec defines the desired state of Algorithm

Type: object

Table 1. Description of Algorithm/v1beta1 spec

Property	Type	Description
spec	object	AlgorithmSpec defines the desired state of Algorithm
spec.description	string	(No Description)
spec.enabled	boolean	(No Description)
spec.manifestBase64	string	(No Description)
spec.name	string	(No Description)
spec.registered	boolean	(No Description)
spec.runtime	string	(No Description)
spec.usedStatus	array	(No Description)
spec.version	string	(No Description)

status

Description: AlgorithmStatus defines the observed state of Algorithm

Type: object

Table 1. Description of Algorithm/v1beta1 status

Property	Type	Description
status	object	AlgorithmStatus defines the observed state of Algorithm

ASM/v1beta1

Create Agile Service Manager deployment. Documentation for additional details regarding install parameters check: Learn More. By installing this product you accept: License Terms

spec

Description: ASMSpec defines the desired state of ASM

Type: object

Table 1. Description of ASM/v1beta1 spec

Property	Type	Description
spec	object	ASMSpec defines the desired state of ASM
spec.advanced	object	Advanced image properties
spec.advanced.imagePullPolicy	string	The default pull policy is IfNotPresent, which causes the Kubelet to skip pulling an image that already exists.
spec.advanced.imagePullRepository	string	Docker registry that all component images are pulled from
spec.asm	object	Topology components
spec.asm.aaionap	boolean	Do you want to deploy AAI ONAP Service
spec.asm.almObserver	boolean	Do you want to deploy ALM Observer
spec.asm.ansibleawxObserver	boolean	Do you want to deploy Ansible AWX Observer
spec.asm.appDisco	object	Application Discovery properties
spec.asm.appDisco.caCertProperty	string	Cert Property
spec.asm.appDisco.caCertSecret	string	Cert Secret
spec.asm.appDisco.caKeyProperty	string	Key Property
spec.asm.appDisco.certSecret	string	DB2 Certificate Secret
spec.asm.appDisco.dbsecret	string	DB2 secret
spec.asm.appDisco.dburl	string	DB2 host to connect
spec.asm.appDisco.enabled	boolean	Enable Application Discovery and Application Discovery Observer
spec.asm.appDisco.scaleDS	integer	(No Description)
spec.asm.appDisco.scaleSSS	integer	(No Description)
spec.asm.appDisco.secure	boolean	Secure DB connection
spec.asm.appDisco.tlsSecret	string	(No Description)
spec.asm.appdynamicsObserver	boolean	Do you want to deploy AppDynamics Observer
spec.asm.awsObserver	boolean	Do you want to deploy AWS Observer
spec.asm.azureObserver	boolean	Do you want to deploy Microsoft Azure Observer
spec.asm.bigcloudfabricObserver	boolean	Do you want to deploy BigCloud Fabric Observer
spec.asm.bigfixinventoryObserver	boolean	Do you want to deploy BigFix Inventory Observer
spec.asm.cienablueplanetObserver	boolean	Do you want to deploy Ciena Blue Planet Observer
spec.asm.ciscoaciObserver	boolean	Do you want to deploy Cisco ACI Observer
spec.asm.contrailObserver	boolean	Do you want to deploy Juniper Contrail Observer
spec.asm.datadogObserver	boolean	Do you want to deploy Datadog Observer
spec.asm.dnsObserver	boolean	Do you want to deploy DNS Observer
spec.asm.dockerObserver	boolean	Do you want to deploy Docker Observer
spec.asm.dynatraceObserver	boolean	Do you want to deploy Dynatrace Observer
spec.asm.fileObserver	boolean	Do you want to deploy File Observer
spec.asm.gitlabObserver	boolean	Do you want to deploy GitLab Observer
spec.asm.googlecloudObserver	boolean	Do you want to deploy Google Cloud Platform Observer

Property	Type	Description
spec.asm.hpnfvdObserver	boolean	Do you want to deploy HPNFVD Observer
spec.asm.iafCartridgeRequirementsName	string	IAF CartridgeRequirements Instance Name for connection to IAF ElasticSearch
spec.asm.iafElasticBindingSecret	string	IAF ElasticSearch Binding Secret for connection to IAF ElasticSearch
spec.asm.iafKafkaBindingSecret	string	IAF Kafka Binding Secret for connection to IAF Kafka
spec.asm.ibmcloudObserver	boolean	Do you want to deploy IBM Cloud Observer (default = false)
spec.asm.itnmObserver	boolean	Do you want to deploy ITNM Observer
spec.asm.jenkinsObserver	boolean	Do you want to deploy Jenkins Observer
spec.asm.junipercsoObserver	boolean	Do you want to deploy Juniper Networks CSO Observer
spec.asm.kubernetesObserver	boolean	Do you want to deploy Kubernetes Observer (default = true)?
spec.asm.netDisco	boolean	Do you want to deploy Network Discovery and Network Discovery Observer
spec.asm.newrelicObserver	boolean	Do you want to deploy New Relic Observer
spec.asm.openstackObserver	boolean	Do you want to deploy OpenStack Observer
spec.asm.rancherObserver	boolean	Do you want to deploy Rancher Observer
spec.asm.restObserver	boolean	Do you want to deploy REST Observer
spec.asm.sdconapObserver	boolean	Do you want to deploy SDC ONAP Observer
spec.asm.servicenowObserver	boolean	Do you want to deploy ServiceNow Observer
spec.asm.sevoneObserver	boolean	Do you want to deploy SevOne Observer
spec.asm.taddmObserver	boolean	Do you want to deploy TADDM Observer
spec.asm.userServiceUrl	string	CEM User Service Url connection
spec.asm.viptelaObserver	boolean	Do you want to deploy Viptela Observer
spec.asm.vmvcenterObserver	boolean	Do you want to deploy VMware VCenter Observer
spec.asm.vmwarensxObserver	boolean	Do you want to deploy VMware NSX Observer
spec.asm.zabbixObserver	boolean	Do you want to deploy Zabbix Observer
spec.clusterDomain	string	Use the fully qualified domain name (FQDN) to specify the cluster domain: apps. ...com
spec.deploymentType	string	Deployment type (small or large)
spec.entitlementSecret	string	Entitlment secret to pull images
spec.helmValuesASM	object	(Not shown within UI)
spec.ingress	object	Vanilla Kube (Not shown within UI)
spec.ingress.port	string	(No Description)
spec.ingress.prefixWithReleaseName	boolean	(No Description)
spec.ingress.tlsSecret	string	(No Description)
spec.license	object	License properties
spec.license.accept	boolean	If you have purchased IBM Cloud Pak for AIOps, you must read and accept this license: https://ibm.biz/cp4aiops-421-license.
spec.persistence	object	Persistence properties
spec.persistence.enabled	boolean	Enable persistence storage
spec.persistence.storageClass	string	Storage Class
spec.persistence.storageSizeCassandraBackup	string	Storage Size Cassandra Backup
spec.persistence.storageSizeCassandraData	string	Storage Size Cassandra Data
spec.persistence.storageSizeElastic	string	Storage Size Elasticsearch
spec.persistence.storageSizeFileObserver	string	Storage size File Observer .
spec.persistence.storageSizeKafka	string	Storage Size Kafka
spec.persistence.storageSizeSdconapObserver	string	Storage Size SDCONAP Observer
spec.persistence.storageSizeZookeeper	string	Storage Size Zookeeper
spec.version	string	Version
spec.zen	object	The Zen Parameters
spec.zen.deploy	boolean	Flag to deploy our own cpd in the same namespace as aimanager
spec.zen.ignoreReady	boolean	Flag to deploy zen customization even if not in ready state
spec.zen.instanceId	string	ID of Zen Service Instance
spec.zen.serviceInstanceName	string	Name of the ZenService Instance object

Property	Type	Description
spec.zen.serviceNamespace	string	Namespace of the ZenService Instance
spec.zen.storage	object	The Zen storage options
spec.zen.storage.storageClassNam e	string	The StorageClassName

status

Description: Observed state of the installed full Cloud deployment.

Type: object

Table 1. Description of ASM/v1beta1 status

Property	Type	Description
status	object	Observed state of the installed full Cloud deployment.
status.managedResources	array	Resources managed by Netcool Agile Service Manager
status.message	string	(No Description)
status.phase	string	Current phase of installed full Cloud deployment.
status.versions	object	Information on versions for Agile Service Manager
status.versions.available	object	(No Description)
status.versions.available.versions	array	Possible available versions for Agile Service Manager
status.versions.reconciled	string	Current version of the reconciled instance

BaseUI/v1

For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.

spec

Description: (No Description)

Type: object

Table 1. Description of BaseUI/v1 spec

Property	Type	Description
spec	object	(No Description)
spec.configuration	object	Additional configuration for the UI deployment
spec.configuration.topologyM odel	string	TopologyModel - Specify the terminology to use in IBM Cloud Pak for AIOps when defining collections of topology resource groups. Valid values are "application" or "service". If this setting is not specified, the default value will be "application".
spec.container	object	ContainerSpec - possible configurations around containers
spec.container.containerPort	integer	(No Description)
spec.container.env	array	(No Description)
spec.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.resources.clai ms	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.resources.limit s	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.resources.requ ests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.image	object	ImageSpec - specification around how to pull an image.
spec.image.path	string	(No Description)

Property	Type	Description
spec.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.image.pullSecret	string	(No Description)
spec.license	object	License - license spec determines whether a license has been accepted
spec.license.accept	boolean	License acceptance switch
spec.name	string	Name of the service being deployed
spec.replicas	integer	(No Description)
spec.size	string	(No Description)
spec.tlsSecretName	string	(No Description)
spec.version	string	Version of the csv deploying
spec.zenserviceName	string	name of the zenservice cr that is used for this deployment

status

Description: (No Description)

Type: object

Table 1. Description of BaseUI/v1 status

Property	Type	Description
status	object	(No Description)
status.conditions	array	(No Description)
status.managedResources	array	(No Description)
status.phase	string	(No Description)
status.versions	object	(No Description)
status.versions.reconciled	string	(No Description)

BundleManifest/v1alpha1

BundleManifest is the Schema for the bundlemanifests API

spec

Description: BundleManifestSpec defines the desired state of BundleManifest

Type: object

Table 1. Description of BundleManifest/v1alpha1 spec

Property	Type	Description
spec	object	BundleManifestSpec defines the desired state of BundleManifest
spec.instanced	object	GitAppSpec defines the desired state of GitApp
spec.instanced.authSecret	object	AuthSecret is the name of the secret containing credentials for Basic-Auth in the username and password keys
spec.instanced.authSecret.name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?
spec.instanced.branch	string	Branch to checkout
spec.instanced.certSecret	object	CertSecret is the name of the secret containing the ca.crt needed to access the repo if the origin uses a self-signed certificate
spec.instanced.certSecret.name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?
spec.instanced.components	array	(No Description)
spec.instanced.repo	string	Repo containing deployment files
spec.prereqs	object	GitAppSpec defines the desired state of GitApp
spec.prereqs.authSecret	object	AuthSecret is the name of the secret containing credentials for Basic-Auth in the username and password keys
spec.prereqs.authSecret.name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?

Property	Type	Description
spec.prereqs.branch	string	Branch to checkout
spec.prereqs.certSecret	object	CertSecret is the name of the secret containing the ca.crt needed to access the repo if the origin uses a self-signed certificate
spec.prereqs.certSecret.name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?
spec.prereqs.components	array	(No Description)
spec.prereqs.repo	string	Repo containing deployment files

status

Description: CommonStatus that can be used with StatusUpdater

Type: object

Table 1. Description of BundleManifest/v1alpha1 status

Property	Type	Description
status	object	CommonStatus that can be used with StatusUpdater
status.conditions	array	(No Description)
status.observedGeneration	integer	(No Description)
status.phase	string	(No Description)
status.resources	object	Resources contained in status

BundleManifest/v1beta1

BundleManifest is the Schema for the bundlemanifests API

spec

Description: BundleManifestSpec defines the desired state of BundleManifest

Type: object

Table 1. Description of BundleManifest/v1beta1 spec

Property	Type	Description
spec	object	BundleManifestSpec defines the desired state of BundleManifest
spec.instanced	object	GitAppSpec defines the desired state of GitApp
spec.instanced.authSecret	object	AuthSecret is the name of the secret containing credentials for Basic-Auth in the username and password keys
spec.instanced.authSecret.name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?
spec.instanced.branch	string	Branch to checkout
spec.instanced.certSecret	object	CertSecret is the name of the secret containing the ca.crt needed to access the repo if the origin uses a self-signed certificate
spec.instanced.certSecret.name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?
spec.instanced.components	array	(No Description)
spec.instanced.repo	string	Repo containing deployment files
spec.prereqs	object	GitAppSpec defines the desired state of GitApp
spec.prereqs.authSecret	object	AuthSecret is the name of the secret containing credentials for Basic-Auth in the username and password keys
spec.prereqs.authSecret.name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?
spec.prereqs.branch	string	Branch to checkout
spec.prereqs.certSecret	object	CertSecret is the name of the secret containing the ca.crt needed to access the repo if the origin uses a self-signed certificate

Property	Type	Description
spec.prereqs.certSecret.name	string	Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?
spec.prereqs.components	array	(No Description)
spec.prereqs.repo	string	Repo containing deployment files

status

Description: CommonStatus that can be used with StatusUpdater

Type: object

Table 1. Description of BundleManifest/v1beta1 status

Property	Type	Description
status	object	CommonStatus that can be used with StatusUpdater
status.conditions	array	(No Description)
status.observedGeneration	integer	(No Description)
status.phase	string	(No Description)
status.resources	object	Resources contained in status

ConnectorUI/v1

For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.

spec

Description: CommonSpecs - specs that are common to determining a deployment.

Type: object

Table 1. Description of ConnectorUI/v1 spec

Property	Type	Description
spec	object	CommonSpecs - specs that are common to determining a deployment.
spec.configuration	object	Additional configuration for the UI deployment
spec.configuration.topologyModel	string	TopologyModel - Specify the terminology to use in IBM Cloud Pak for AIOps when defining collections of topology resource groups. Valid values are "application" or "service". If this setting is not specified, the default value will be "application".
spec.container	object	ContainerSpec - possible configurations around containers
spec.container.containerPort	integer	(No Description)
spec.container.env	array	(No Description)
spec.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.image	object	ImageSpec - specification around how to pull an image.
spec.image.path	string	(No Description)
spec.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.image.pullSecret	string	(No Description)

Property	Type	Description
spec.license	object	License - license spec determines whether a license has been accepted
spec.license.accept	boolean	License acceptance switch
spec.name	string	Name of the service being deployed
spec.replicas	integer	(No Description)
spec.size	string	(No Description)
spec.tlsSecretName	string	(No Description)
spec.version	string	Version of the csv deploying
spec.zenserviceName	string	name of the zenservice cr that is used for this deployment

status

Description: (No Description)

Type: object

Table 1. Description of ConnectorUI/v1 status

Property	Type	Description
status	object	(No Description)
status.conditions	array	(No Description)
status.managedResources	array	(No Description)
status.phase	string	(No Description)
status.versions	object	(No Description)
status.versions.reconciled	string	(No Description)

InsightsUI/v1

For documentation regarding install parameters, see IBM Documentation. By installing this product you accept the license terms.

spec

Description: (No Description)

Type: object

Table 1. Description of InsightsUI/v1 spec

Property	Type	Description
spec	object	(No Description)
spec.asm	object	Contact details for ASM
spec.asm.bindingSecret	string	Data store binding secret
spec.configuration	object	Additional configuration for the UI deployment
spec.configuration.topologyModel	string	TopologyModel - Specify the terminology to use in IBM Cloud Pak for AIOps when defining collections of topology resource groups. Valid values are "application" or "service". If this setting is not specified, the default value will be "application".
spec.container	object	ContainerSpec - possible configurations around containers
spec.container.containerPort	integer	(No Description)
spec.container.datarouter	object	(No Description)
spec.container.datarouter.containerPort	integer	(No Description)
spec.container.datarouter.env	array	(No Description)
spec.container.datarouter.image	object	ImageSpec - specification around how to pull an image.
spec.container.datarouter.image.path	string	(No Description)
spec.container.datarouter.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.container.datarouter.image.pullSecret	string	(No Description)

Property	Type	Description
spec.container.datarouter.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.datarouter.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.datarouter.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.datarouter.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.datarouter.tlsSecretName	string	(No Description)
spec.container.env	array	(No Description)
spec.container.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.ui	object	(No Description)
spec.container.ui.containerPort	integer	(No Description)
spec.container.ui.env	array	(No Description)
spec.container.ui.image	object	ImageSpec - specification around how to pull an image.
spec.container.ui.image.path	string	(No Description)
spec.container.ui.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.container.ui.image.pullSecret	string	(No Description)
spec.container.ui.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.ui.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.ui.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.ui.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.ui.tlsSecretName	string	(No Description)
spec.container.uiApi	object	(No Description)
spec.container.uiApi.containerPort	integer	(No Description)

Property	Type	Description
spec.container.uiApi.env	array	(No Description)
spec.container.uiApi.image	object	ImageSpec - specification around how to pull an image.
spec.container.uiApi.image.path	string	(No Description)
spec.container.uiApi.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.container.uiApi.image.pullSecret	string	(No Description)
spec.container.uiApi.resources	object	ResourceRequirements describes the compute resource requirements.
spec.container.uiApi.resources.claims	array	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.
This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.		
This field is immutable.		
spec.container.uiApi.resources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiApi.resources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
spec.container.uiApi.tlsSecretName	string	(No Description)
spec.demodatasets	array	(No Description)
spec.elasticSearch	object	Contact details for ElasticSearch
spec.elasticSearch.bindingSecret	string	Data store binding secret
spec.image	object	ImageSpec - specification around how to pull an image.
spec.image.path	string	(No Description)
spec.image.pullPolicy	string	PullPolicy describes a policy for if/when to pull a container image
spec.image.pullSecret	string	(No Description)
spec.kafka	object	Contact details for Kafka
spec.kafka.bindingSecret	string	Data store binding secret
spec.license	object	License - license spec determines whether a license has been accepted
spec.license.accept	boolean	License acceptance switch
spec.name	string	Name of the service being deployed
spec.postgres	object	Contact details for postgres
spec.postgres.bindingSecret	string	Data store binding secret
spec.replicas	integer	(No Description)
spec.size	string	(No Description)
spec.tlsSecretName	string	(No Description)
spec.version	string	Version of the csv deploying
spec.zenserviceName	string	name of the zenservice cr that is used for this deployment

status

Description: (No Description)

Type: object

Table 1. Description of InsightsUI/v1 status

Property	Type	Description
status	object	(No Description)
status.conditions	array	(No Description)
status.managedResources	array	(No Description)
status.phase	string	(No Description)
status.versions	object	(No Description)
status.versions.reconciled	string	(No Description)

Installation/v1alpha1

Leverage advanced, explainable AI to resolve incidents faster, gain visibility into critical workloads, and intelligently manage key application resources. Learn More. By installing this product you accept the license terms.

spec

Description: For detailed information about the following settings, see the IBM Documentation at https://ibm.biz/cp4aiops_421_install

Type: object

Table 1. Description of Installation/v1alpha1 spec

Property	Type	Description
spec	object	For detailed information about the following settings, see the IBM Documentation at https://ibm.biz/cp4aiops_421_install
spec.automationFoundation	object	AutomationFoundation contains parameters that can be used to customize how the IBM Automation Foundation objects are configured.
spec.automationFoundation.automationBase	string	AutomationBase is the name of the IBM Automation Foundation "AutomationBase" CR instance that the Cloud Pak should use. If a name is not specified, and no "AutomationBase" instances exist in the namespace, the Cloud Pak will attempt to provision one.
spec.automationFoundation.automationUIConfig	string	AutomationUIConfig is the name of the IBM Automation Foundation "AutomationUIConfig" CR instance that the Cloud Pak should use. If a name is not specified, and no "AutomationUIConfig" instances exist in the namespace, the Cloud Pak will attempt to provision one.
spec.automationFoundation.enabled	boolean	The Enabled flag controls whether the Cloud Pak will register with IBM Automation Foundation. The default value is 'true'. When the flag is set to 'false', the Cloud Pak will not create any Automation Foundation objects, including those referenced in other parts of the AutomationFoundation spec.
spec.enableConnectionModule	boolean	Use the Secure Tunnel to safely connect to remote data sources.
spec.imagePullSecret	string	Select the image pull secret to use for the Cloud Pak. If no secret is selected, the global pull secret will be used.
spec.installPlanApproval	string	Approval is the user approval policy for an InstallPlan. It must be one of "Automatic" or "Manual".
spec.license	object	License acceptance is required to create resources.
spec.license.accept	boolean	I accept the product license after reading it at https://ibm.biz/cp4aiops-421-license
spec.pakModules	array	Pak Modules are modular pieces that may be enabled for deployment
spec.proxyServer	string	If the target cluster is air-gapped, specify the address and port number of an HTTP proxy that will provide access to the Slack or MS Teams service used for ChatOps, for example, proxy.company.com:8080.
spec.resourceOverrides	string	Specify the name of a configmap that finely tunes the capabilities and capacity of an installation.
spec.size	string	Select a size to determine the amount of virtual resources requested and the level of high availability support.
spec.storageClass	string	Select a file storage class that supports ReadWriteMany access mode.
spec.storageClassLargeBlock	string	Select a storage class with a large block size (for example, 64k).
spec.topologyModel	string	Specify the terminology to use in CP4AIOps when defining collections of topology resource groups. Valid values are "application" or "service". Default value will be "application".

status

Description: InstallationStatus defines the observed state of Installation

Type: object

Table 1. Description of Installation/v1alpha1 status

Property	Type	Description
status	object	InstallationStatus defines the observed state of Installation
status.componentnamespace	string	(No Description)
status.componentstatus	object	(No Description)
status.customProfileConfigmap	string	(No Description)
status.customProfileValidationStatus	string	(No Description)

Property	Type	Description
status.failureMessage	string	(No Description)
status.failureReason	string	(No Description)
status.imagePullSecret	string	(No Description)
status.licenseacceptance	string	LicenseAcceptanceStatus describes the status of the license acceptance
status.locations	object	(No Description)
status.locations.cloudPakUiUrl	string	Launch the Cloud Pak
status.locations.csAdminHubUrl	string	Manage services and processes
status.phase	string	InstallationPhase describes the phase of the overall Installation
status.size	string	(No Description)
status.storageclass	string	(No Description)
status.storageclasslargeblock	string	(No Description)

Kong/v1alpha1

Kong is the Schema for kong gateway

status

Description: KongStatus defines the observed state of Kong, uneditable

Type: object

Table 1. Description of Kong/v1alpha1 status

Property	Type	Description
status	object	KongStatus defines the observed state of Kong, uneditable
status.conditions	array	(No Description)

Tunnel/v1

Tunnel is the Schema for the tunnels API

spec

Description: SecureTunnelSpec defines the desired state of Tunnel

Type: object

Table 1. Description of Tunnel/v1 spec

Property	Type	Description
spec	object	SecureTunnelSpec defines the desired state of Tunnel
spec.ApiServerURL	string	(No Description)
spec.apiReplicas	integer	(No Description)
spec.apiServerURL	string	(No Description)
spec.applicationMappingPortRang	string	(No Description)
spec.auditReceiver	object	(No Description)
spec.auditReceiver.mongoDB	array	(No Description)
spec.auditReceiver.syslog	array	(No Description)
spec.auditReceivers	array	(No Description)
spec.commonServiceNamespace	string	(No Description)
spec.controllerReplicas	integer	(No Description)

Property	Type	Description
spec.defaultTunnelWorkerResources	object	(No Description)
spec.defaultTunnelWorkerResources.limits	object	(No Description)
spec.defaultTunnelWorkerResources.limits.cpu	string	(No Description)
spec.defaultTunnelWorkerResources.limits.memory	string	(No Description)
spec.defaultTunnelWorkerResources.requests	object	(No Description)
spec.defaultTunnelWorkerResources.requests.cpu	string	(No Description)
spec.defaultTunnelWorkerResources.requests.memory	string	(No Description)
spec.iamEndpoint	string	(No Description)
spec.imageDigest	string	(No Description)
spec.imageRepo	string	(No Description)
spec.imageTag	string	(No Description)
spec.ingressClass	string	(No Description)
spec.ingressDomain	string	(No Description)
spec.runAsStandAlone	boolean	(No Description)
spec.enableEgressNetworkPolicy	boolean	(No Description)
spec.enableAdminLogin	boolean	(No Description)
spec.connectionTLSCertDurationTime	string	(No Description)
spec.inventory	object	(No Description)
spec.inventory.inventoryPort	integer	(No Description)
spec.inventory.inventoryService	string	(No Description)
spec.inventory.namespace	string	(No Description)
spec.inventory.searchApiSecrets	string	(No Description)
spec.jwtPublicURL	string	(No Description)
spec.kubernetesEvn	boolean	(No Description)
spec.license	object	Opt-in license acceptance is required to create resources
spec.license.accept	boolean	(No Description)
spec.noManagementUI	boolean	(No Description)
spec.pullSecret	string	(No Description)
spec.runType	string	(No Description)
spec.showInConsole	boolean	(No Description)
spec.showInContainerLog	boolean	(No Description)
spec.size	string	(No Description)
spec.tunnelWorkerResources	object	////

Property	Type	Description
spec.tunnelWorkerResources.limits	object	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/
spec.tunnelWorkerResources.requests	object	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. More info: https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/
spec.uiImageDigest	string	(No Description)
spec.uiImageRepo	string	(No Description)
spec.uiImageTag	string	(No Description)
spec.uiReplicas	integer	(No Description)
spec.useTag	boolean	(No Description)
spec.version	string	(No Description)
spec.workerReplicas	integer	(No Description)
spec.zenCoreCertificateName	string	(No Description)

status

Description: SecureTunnelStatus defines the observed state of Tunnel

Type: object

Table 1. Description of Tunnel/v1 status

Property	Type	Description
status	object	SecureTunnelStatus defines the observed state of Tunnel
status.conditions	array	(No Description)
status.versions	object	INSERT ADDITIONAL STATUS FIELD - define observed state of cluster Important: Run "operator-sdk generate k8s" to regenerate code after modifying this file Add custom validation using kubebuilder tags: https://book-v1.book.kubebuilder.io/beyond_basics/generating_crd.html
status.versions.description	string	(No Description)
status.versions.reconciled	string	(No Description)
status.versions.status	string	(No Description)
status.versions.tenantID	string	(No Description)

Infrastructure Automation custom resource definitions

Use this information to learn about the custom resource definitions (CRDs) used by Infrastructure Automation.

In the following table, CRDs that do not have a hyperlink must not be manually edited unless otherwise instructed, and are manipulated only by the operator.

Table 1. Infrastructure Automation CRDs

Kind	API version	Operator name	Description
IAConfig	v1alpha1	ibm-infrastructure-automation-operator	Build, deploy, and manage VMs and Kubernetes clusters. Enable self-service orchestration of resources across clouds. Learn More.
IAManageService	v1alpha1	ibm-management-cam-install	ManageService is the Schema for the manageservices API
ManageIQ	v1alpha1	ibm-management-im-install	ManageIQ is the Schema for the manageiqs API
IMInstall	v1alpha1	ibm-management-im-install	IMInstall is the Schema for the iminstalls API

IAConfig/v1alpha1

The **`IAConfig`** custom resource definition (CRD) is used to configure and install an instance of Infrastructure Automation.

Build, deploy, and manage VMs and Kubernetes clusters. Enable self-service orchestration of resources across clouds. Learn More.

spec

Description: IAConfigSpec defines the desired state of IAConfig

Type: object

Table 1. Description of IAConfig/v1alpha1 spec

Property	Type	Description
spec	object	IAConfigSpec defines the desired state of IAConfig
spec.imagePullSecret	string	Select a secret to serve as the ImagePullSecret
spec.infraAutoComposableComponents	array	Enable Infrastructure management (im) and/or Manage services (cam).
spec.license	object	License acceptance is required to create resources.
spec.license.accept	boolean	I accept the product license after reading it at https://ibm.biz/cp4aiops-421-license
spec.skipAutomationBaseStatus	boolean	This field is reserved for support person to use.
spec.skipAutomationUiConfigStatus	boolean	This field is reserved for support person to use.
spec.storageClass	string	Select a storage class that supports ReadWriteMany access mode.
spec.storageClassLargeBlock	string	Select a storage class with a large block size (for example, 64k).

Important: The following fields are deprecated and should not be used in Infrastructure Automation 4.2.1:

- skipAutomationBaseStatus
- skipAutomationUiConfigStatus

status

Description: IAConfigStatus defines the observed state of IAConfig

Type: object

Table 2. Description of IAConfig/v1alpha1 status

Property	Type	Description
status	object	IAConfigStatus defines the observed state of IAConfig
status.failureMessage	string	INSERT ADDITIONAL STATUS FIELD - define observed state of cluster Important: Run "make" to regenerate code after modifying this file
status.phase	string	InstallationPhase describes the phase of the overall Installation
status.storageclass	string	(No Description)
status.storageclasslargeblock	string	(No Description)
status.version	string	(No Description)

ManageService/v1

ManageService is the Schema for the manageservices API.

spec

Description: ManageServiceSpec defines the desired state of ManageService.

Type: object

Table 1. Description of IAManageService spec

Property	Type	Description	Default

Property	Type	Description	Default
camAPI	object	Configuration for Cloud Automation Manager API microservices	`{"camComposerApi":{"resources":{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}},"camComposerUi":{"resources":{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}},"camIaas":{"resources":{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}},"camMongo":{"resources":{"limits":{"cpu":"500m","memory":"4Gi"},"requests":{"cpu":"100m","memory":"2Gi"}}},"camOrchestration":{"resources":{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}},"camPortalUi":{"resources":{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}},"camProviderTerraformApi":{"resources":{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}},"terraformSSHDir":{"terraformSSHDirSecret":""}},"camSecret":{"secretName":"cam-api-secret"},"camServiceLibraryUi":{"resources":{"limits":{"cpu":"300m","memory":"512Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}},"camServiceLibraryUiAPi":{"resources":{"limits":{"cpu":"100m","memory":"512Mi"},"requests":{"cpu":"50m","memory":"256Mi"}}},"camTenantApi":{"resources":{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}},"camUiBasic":{"resources":{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}},"certificate":{"certName":"cert"},"replicaCount":1}`
camAPI.replicaCount	int	Number of replicas for cam microservices	`1`
camController.replicaCount	int	Number of replicas for Controller	`1`
camLoggingPolicies	object	Configuration for the logging level used with Cloud Automation Manager microservices	`{"logLevel":"info"}`
camLoggingPolicies.logLevel	string	Log Level for logging	`"info"`
camMongoPV	object	MongoDB Persistant Storage	`{"name":"cam-mongo-pv","persistence":{"accessMode":"ReadWriteOnce","enabled":true,"existingClaimName":"","size":"15Gi","storageClassName":"","useDynamicProvisioning":false}}`
camMongoPV.persistence.accessMode	string	Access Mode for Storage	`"ReadWriteOnce"`
camMongoPV.persistence.enabled	bool	Enable persistance	`true`
camMongoPV.persistence.existingClaimName	string	Specify the name of the Existing Claim to be used by your application, empty string means don't use an existClaim	`""`
camMongoPV.persistence.size	string	MongoDB PVC size (in Gi)	`"15Gi"`
camMongoPV.persistence.storageClassName	string	Specify the name of the StorageClass empty string means don't use a StorageClass	`""`

Property	Type	Description	Default
camMongoPV.persistence.useDynamicProvisioning	bool	useDynamicProvisioning for persistence	`false`
camProviderTerraform	object	Configuration for Cloud Automation Manager Provider Terraform	`{"isolateRuntime":false,"replicaCount":1,"runtime":{"replicaCount":0},"terraformSSHDir":{"terraformSSHDirSecret":""}}`
camProviderTerraform.isolateRuntime	bool	Select to enable isolation mode for Provider Terraform runtime. In this mode, the runtime runs in separate pods and have read only access to the cam-terraform-pv.	`false`
camProviderTerraform.replicaCount	int	Number of replicas for camProviderTerraformApi	`1`
camProviderTerraform.runtime	object	Configuration for Cloud Automation Manager Provider Terraform runtime	`{"replicaCount":0}`
camProviderTerraform.runtime.replicaCount	int	Number of replicas for Provider Terraform Runtime. This value is only applicable when camProviderTerraform.isolateRuntime is enabled	`0`
camProviderTerraform.terraformSSHDir	object	Cloud Automation Manager Provider Terraform SSH Directory	`{"terraformSSHDirSecret":""}`
camProviderTerraform.terraformSSHDir.terraformSSHDirSecret	string	Secret name for Cloud Automation Manager Provider Terraform SSH Directory Contents	`""`
camProxy	object	CAM Proxy configuration	`{"customNginxSSLCertificate":{"customNginxSSLCertName":""},"limits":{"cpu":"500m","memory":"1500Mi"},"replicaCount":1,"requests":{"cpu":"100m","memory":"256Mi"},"resources":null}`
camProxy.customNginxSSLCertificate.customNginxSSLCertName	string	Nginx SSL Custom Certificate Secret Name	`""`

Property	Type	Description	Default
camProxy.limits.cpu	string	The CPU limit	`"500m"`
camProxy.limits.memory	string	The memory limit	`"1500Mi"`
camProxy.replicaCount	int	Number of replicas for Proxy	`1`
camProxy.requests.cpu	string	The requested CPU	`"100m"`
camProxy.requests.memory	string	The requested memory	`"256Mi"`
camUI	object	UI Configuration	`{"camUISecret":{"secretName":"cam-ui-secret","sessionKey":"opsConsole.sid","sessionTTL":"43200"},"replicaCount":1}`
camUI.camUISecret	object	User Interface Secret	`{"secretName":"cam-ui-secret","sessionKey":"opsConsole.sid","sessionTTL":"43200"}`
camUI.camUISecret.secretName	string	User interface Secret Name	`"cam-ui-secret"`
camUI.camUISecret.sessionKey	string	Session Key	`"opsConsole.sid"`
camUI.camUISecret.sessionTTL	string	Session timeout in seconds	`"43200"`
camUI.replicaCount	int	Number of replicas for UI	`1`
database	object	Optionally use an external MongoDB	`{"bundled":true}`
database.bundled	bool	Use internal MongoDB. Set to false for external. For more information, see Using an external MongoDB in the IBM Knowledge Center	`true`
license	object	License acceptance	`{"accept":true}`
license.accept	bool	Accept the license to use the product"	`true`
proxy	object	Configuration for a proxy used with Cloud Automation Manager. Supply details in the secure values secret.	`{"useProxy":false}`
proxy.useProxy	bool	Select to use a proxy with CAM and fill out the fields in the Secure Values secret (see the Knowledge Center install instructions for more details)	`false`

Property	Type	Description	Default
resources	object	Amount of memory and CPU to assign to Cloud Automation Manager microservices	`{"limits":{"cpu":"500m","memory":"1500Mi"},"requests":{"cpu":"100m","memory":"256Mi"}}`
resources.limits.cpu	string	The memory limit	`"500m"`
resources.limits.memory	string	The CPU limit	`"1500Mi"`
resources.requests.cpu	string	The requested CPU	`"100m"`
resources.requests.memory	string	The requested memory	`"256Mi"`
roks	bool	Select to indicate it is IBM Cloud ROKS System	`false`
roksRegion	string	IBM ROKS Storage Region selection	`""`
roksZone	string	IBM ROKS Storage Zone selection	`""`

IMInstall/v1alpha1

IMInstall is the Schema for the iminstalls API

spec

Description: IMInstallSpec defines the desired state of IMInstall

Type: object

Table 1. Description of IMInstall/v1alpha1 spec

Property	Type	Description
spec	object	IMInstallSpec defines the desired state of IMInstall
spec.appAnnotations	object	Optional Annotations to apply to the Httpd, Kafka, Memcached, Orchestrator and PostgresQL Pods
spec.appName	string	Application name used for deployed objects (default: iminstall)
spec.applicationDomain	string	Domain name for the external route. Used for external authentication configuration
spec.applicationImage	string	Image string used for the infrastructure management application deployment. By default it is based on the olm.relatedImage.application annotation
spec.backupLabelName	string	This label will be applied to essential resources that need to be backed up (default: manageiq.org/backup)
spec.baseWorkerImage	string	Image string used for the base worker deployments By default this is determined by the orchestrator pod
spec.databaseRegion	string	Database region number (default: 0)
spec.databaseSecret	string	Secret containing the database access information, content generated if not provided (default: postgresql-secrets)
spec.databaseVolumeCapacity	string	Database volume size (default: 50Gi)
spec.deployMessagingService	boolean	(Deprecated) Flag to indicate if Kafka and Zookeeper should be deployed (default: false)
spec.enableApplicationLocalLogin	boolean	Flag to allow logging into the application without SSO (default: false)

Property	Type	Description
spec.enableSSO	boolean	Flag to enable SSO in the application (default: false)
spec.enforceWorkerResourceConstraints	boolean	Flag to trigger worker resource constraint enforcement (default: true)
spec.httpdAuthConfig	string	Secret containing the httpd configuration files Mutually exclusive with the OIDCClientSecret and OIDCProviderURL if using openid-connect
spec.httpdAuthenticationType	string	Type of httpd authentication (default: internal) Options: internal, external, active-directory, saml, openid-connect Note: external, active-directory, and saml require an httpd container with elevated privileges
spec.httpdCpuLimit	string	Httpd deployment CPU limit (default: 1000m)
spec.httpdCpuRequest	string	Httpd deployment CPU request (default: 50m)
spec.httpdImage	string	Image string used for the httpd deployment (default: os.Getenv("HTTPD_IMAGE"))
spec.httpdImageNamespace	string	Deprecated: Image namespace used for the httpd deployment (default: docker-na-public.artifactory.swg-devops.com/hyc-cp4mcm-team-docker-local/infra-mgmt) Note: the exact image will be determined by the authentication method selected
spec.httpdImageTag	string	Deprecated: Image tag used for the httpd deployment (default: 2-el8)
spec.httpdMemoryLimit	string	Httpd deployment memory limit (default: 200Mi)
spec.httpdMemoryRequest	string	Httpd deployment memory request (default: 100Mi)
spec.imagePullSecret	string	Secret containing the image registry authentication information needed for the manageiq images
spec.initialAdminGroupName	string	Group name to create with the super admin role. This can be used to seed a group when using external authentication
spec.internalCertificatesSecret	string	Secret containing all of the necessary certificates to secure communication between pods (default: internal-certificates-secret)
spec.kafkaCpuRequest	string	Kafka deployment CPU request (default: 250m)
spec.kafkaCpulimit	string	Kafka deployment CPU limit (default: 250m)
spec.kafkaImage	string	Image string used for the kafka deployment (default: os.Getenv("KAFKA_IMAGE"))
spec.kafkaImageName	string	Deprecated: Image used for the kafka deployment (default: docker.io/bitnami/kafka)
spec.kafkaImageTag	string	Deprecated: Image tag used for the kafka deployment (default: latest)
spec.kafkaMemoryLimit	string	Kafka deployment memory limit (default: 1024Mi)
spec.kafkaMemoryRequest	string	Kafka deployment memory request (default: 256Mi)
spec.kafkaSecret	string	Secret containing the kafka access information, content generated if not provided (default: kafka-secrets)
spec.kafkaVolumeCapacity	string	Kafka volume size (default: 1Gi)
spec.license	object	Opt-in license acceptance is required to create resources
spec.license.accept	boolean	Accept is an opt-in license acceptance required to deploy resources
spec.memcachedCpuLimit	string	Memcached deployment CPU limit (default: 1000m)
spec.memcachedCpuRequest	string	Memcached deployment CPU request (default: 50m)
spec.memcachedImage	string	Image string used for the memcached deployment (default: :)
spec.memcachedImageName	string	Deprecated: Image used for the memcached deployment (default: manageiq/memcached)
spec.memcachedImageTag	string	Deprecated: Image tag used for the memcached deployment (default: 1-el8)
spec.memcachedMaxConnection	string	Memcached max simultaneous connections (default: 1024)
spec.memcachedMaxMemory	string	Memcached item memory in megabytes (default: 64)

Property	Type	Description
spec.memcachedMemoryLimit	string	Memcached deployment memory limit (default: 200Mi)
spec.memcachedMemoryRequest	string	Memcached deployment memory request (default: 100Mi)
spec.memcachedSlabPageSize	string	Memcached max item size (default: 1m, min: 1k, max: 1024m)
spec.migrationsRan	array	A list of CR data migrations that have been run
spec.oidcAuthIntrospectionURL	string	URL for OIDC authentication introspection Only used with the openid-connect authentication type. If not specified, the operator will attempt to fetch its value from the "introspection_endpoint" field in the Provider metadata at the OIDCProviderURL provided.
spec.oidcCaCertSecret	string	Secret containing the trusted CA certificate file(s) for the OIDC server. Only used with the openid-connect authentication type
spec.oidcClientSecret	string	Secret name containing the OIDC client id and secret Only used with the openid-connect authentication type
spec.oidcProviderURL	string	URL for the OIDC provider Only used with the openid-connect authentication type
spec.orchestratorCpuLimit	string	Orchestrator deployment CPU limit (default: 1000m)
spec.orchestratorCpuRequest	string	Orchestrator deployment CPU request (default: 200m)
spec.orchestratorImage	string	Image string used for the orchestrator deployment (default: / :)
spec.orchestratorImageName	string	Deprecated: Image name used for the orchestrator deployment (default: infra-mgmt-orchestrator)
spec.orchestratorImageNamespace	string	Deprecated: Image namespace used for the orchestrator and worker deployments (default: manageiq)
spec.orchestratorImageTag	string	Deprecated: Image tag used for the orchestrator and worker deployments (default: latest-17.y)
spec.orchestratorInitialDelay	string	Number of seconds to wait before starting the orchestrator liveness check (default: 2400)
spec.orchestratorMemoryLimit	string	Orchestrator deployment memory limit (default: 2048Mi)
spec.orchestratorMemoryRequest	string	Orchestrator deployment memory request (default: 1024Mi)
spec.postgresqlCpuLimit	string	PostgreSQL deployment CPU limit (default: 10000m)
spec.postgresqlCpuRequest	string	PostgreSQL deployment CPU request (default: 500m)
spec.postgresqlImage	string	Image string used for the postgresql deployment (default: :)
spec.postgresqlImageName	string	Deprecated: Image used for the postgresql deployment (Default: docker.io/manageiq/postgresql)
spec.postgresqlImageTag	string	Deprecated: Image tag used for the postgresql deployment (Default: 13-el8)
spec.postgresqlMaxConnections	string	PostgreSQL maximum connection setting (default: 1000)
spec.postgresqlMemoryLimit	string	PostgreSQL deployment memory limit (default: 8192Mi)
spec.postgresqlMemoryRequest	string	PostgreSQL deployment memory request (default: 2048Mi)
spec.postgresqlSharedBuffers	string	PostgreSQL shared buffers setting (default: 1GB)
spec.postgresqlVersion	string	PostgreSQL Image version (default: 13)
spec.serverGuid	string	Server GUID (default: auto-generated)
spec.storageClassName	string	StorageClass name that will be used by manageiq data stores
spec.tlsSecret	string	Secret containing the tls cert and key for the ingress, content generated if not provided (default: tls-secret)

Property	Type	Description
spec.uiWorkerImage	string	Image string used for the UI worker deployments By default this is determined by the orchestrator pod
spec.webserverWorkerImage	string	Image string used for the webserver worker deployments By default this is determined by the orchestrator pod
spec.zookeeperCpuRequest	string	Zookeeper deployment CPU request (default: 250m)
spec.zookeeperCpulimit	string	Zookeeper deployment CPU limit (default: 250m)
spec.zookeeperImage	string	Image string used for the zookeeper deployment (default: :)
spec.zookeeperImageName	string	Deprecated: Image used for the zookeeper deployment (default: docker.io/bitnami/zookeeper)
spec.zookeeperImageTag	string	Deprecated: Image tag used for the zookeeper deployment (default: latest)
spec.zookeeperMemoryLimit	string	Zookeeper deployment memory limit (default: 256Mi)
spec.zookeeperMemoryRequest	string	Zookeeper deployment memory request (default: 256Mi)
spec.zookeeperVolumeCapacity	string	Zookeeper volume size (default: 1Gi)

status

Description: IMInstallStatus defines the observed state of IMInstall

Type: object

Table 2. Description of IMInstall/v1alpha1 status

Property	Type	Description
status	object	IMInstallStatus defines the observed state of IMInstall

Accessing APIs

You can access and use IBM Cloud Pak® for AIOps APIs after you obtain an access token for authentication.

About this task

To use any IBM Cloud Pak for AIOps API, you must first authenticate your credentials by including an access token in your API calls. This token is required because a password-based authentication approach is not supported for API calls with IBM Cloud Pak for AIOps, for instance, you cannot call an LDAP server to directly authenticate passwords.

The token that is generated is scoped to your user profile and the permissions that you are assigned when the access token is generated. With your access token, you can use API to access everything that you would typically be able to access when you log in to the IBM Cloud Pak Automation console.

To get a Platform UI access token, first you must have either an Identity and Access Management (IAM) access token or an API key that is authenticated by your user credentials. Then, you can use the IAM access token or API key to obtain your Platform UI access token. This Platform UI access token must be included in the authorization header of your calls to IBM Cloud Pak for AIOps API:

To create your token, complete either of the following procedures based on your preferred method:

- Get access token by using your API key
- Get access token by using your username and password
- Get authentication by using Platform UI API Key

Important: You can directly access the Swagger UI for only the Topology API and Metric API.

- For more information about accessing Metric service Swagger UI, see Accessing Metric service Swagger UI.
- For more information about accessing topology service Swagger UI, see Accessing Topology service Swagger UI.

Get access token by using your API key

1. Generate your platform API key:

    1. Log in to the IBM Cloud Pak Automation console to authenticate your user credentials.

       For more information, such as for how to obtain the URL route for accessing the Automation console, see Accessing the Automation console.

    2. From the toolbar, click your avatar.

    3. Click **Profile and settings**.

    4. Click **API key** > **Generate new key**.

    5. Click **Generate**.

    6. Click **Copy** and save your key somewhere safe. You cannot recover this key if you lose it.

    If you lose your API key, repeat the preceding steps to generate a new API key. Your old API key becomes invalid, and any applications or scripts cannot authenticate to the platform until you provide your new API key. For more information, see Generating API keys for authentication.

2. Get your Platform UI access token by running the following cURL command:

```
curl -k -X POST \
  https://<hostname>/icp4d-api/v1/authorize \
  -H 'Content-Type: application/json' \
  -d '{
    "username": "<username>",
    "api_key": "<API_key>"
  }'
```

    - `<hostname>` - The IBM Cloud Pak platform (`cpd`) route hostname.
    - `<username>` - Your username.
    - `<API_key>` - The API key that you generated in the previous step.

    The API response can resemble the following sample:

```
{
    "_messageCode_": "200",
    "message": "Success",
    "token": "<encoded-token-string>"
}
```

    - `<encoded-token-string>` - Your Platform UI access token.

Get access token by using your username and password

1. Get your Identity and Access Management (IAM) access token with the following cURL command:

```
curl -k -X POST -H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8" \
-d "grant_type=password&username=<username>&password=<password>&scope=openid" \
https://<cluster_address>/idprovider/v1/auth/identitytoken
```

    Where:

    - `<username>` - Your username.
    - `<password>` - Your password.
    - `<cluster_address>` - The IBM Cloud Pak Automation console route. For more information about how to get this Automation console route, see Accessing the Automation console.

    The command returns an `access_token`, `refresh_token`, and `id_token`, as shown in the following example:

```
{"access_token":
"eb837eaf32459b711945a9d2259880119056e805ff0d2f36421cc171f94e58fef349f0005d217e36889b62271d9f00fc7cb5b1f
e5a86546d9dee8e22bca39b8f90d6cb61dae7fc383447823a09e380feeefba5bea0c994408470a49db0df32ddc2b0cca9381519e
60a63daae9b87ebfe9400b0c4af818b7f7d6c32e21465909efc8aa02804808f23ff96ac342b3b1c35230ac8858dbcb7979995d70
44c7b9cb05945c91b63a938703641e0fded339fb4c22e2383743a94a30c41892804193744e0c0f020909f9579555bf691b240faf
b558f76877fe0cb88ecbb3266fedbc7c541129270f67784d11ed658998b536841e0fdfc50a9ad056d2cabf717cb13326e4f620a6
ce172d8da4701b820c5ffe23223e7fb5725b244a1dd45538a0c7ca09a643759aaa2d8585a28689cae968ab3328351e3c38a8b199
040067ca5837169ce62a88282d1c8551d762fbdff77727cb51dd62213cef58dfb88e304abbc48063246b7e9f39650a0ac86f6c72
973b702b79faf34b68afb9412c9e0e56b104e12bf1ea3764faeac258fe1c3e896da412607a71ad8b4224efcfa0eafbc15f5e7af5
b8baa41163c220419c7249e9652ca7b5692b42cbe4c7d88c18d77440ec350582f51880e7354eed76ebd8ce760b27a6ca5808c6fc
51ef843ee5d98e5bbafaeb4096301853fa5fdc876275defd3ecabd0eb656c7cd2441e523e0c5468a1f261fb44",
"token_type":"Bearer","expires_in":43199,"scope":"openid"
"refresh_token":"6q4griAg9yCiGINQvF0Dp7N9hqXhcXZrAsqWWYgl6XQ80Uexsq",
"id_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiYWRmZDc4MmEwOTc1ZTNmMzc2ZTkxZTI3YjJkNTY
xZmQ0OTNiNTQzMSIsInJlYWxtTmFtZSI6ImN1c3RvbVJlYWxtIiwidW5pcXVlU2VjdXJpdHlOYW1lIjoiYWRtaW4iLCJpc3MiOiJodHR
wczovL215Y2x1c3Rlci5pY3A6OTQ0My9vaWRjL2VuZHBvaW50L09QIiwiYXVkIjoiMGQzYzA3MTc5OTYxYmEzMWEyODY5NDU0NDQwM2E
0NDYiLCJleHAiOjE1NTQ5MTQ2NTIsImlhdCI6MTU1NDg4NTg1Miwic3ViIjoiYWRtaW4iLCJ0ZWFtUm9sZU1hcHBpbmdzIjpbXX0.CnT
```

```
0qWECpJR9R16W-IOqrXjSJR8DelRsDUXcX6hy_I0DPQ7hU55Bhcq6UChEg3qiWWRbKwrFIxikXPjEjw2B9oziEd8U8AEO-
4LEaXOpc5Lk1shvyxBQFDDgyUwgyGb-erRbO_Sl1K4xotuTLg4nhoydwTXs7lZn7GC4UW8j1qkhlbFe5iLgKidCZsjyPo-
2GNYEQn0ufHH3KCR4DkHi6GX2RUxisNecwDzNl9P5JSyjlS-
r5QUZJ0b0DytKuY5HxpswpIFaO9U8JlYAFoOZ18eO_CzERHRQ_Ii1ePmagGAk-
eLJjmCNqY1zynfpEUuKlWUR5rVGHGzSbGA8J4CLvg"}
```

From this example, the following content is the **access token** that is needed for obtaining a platform UI access token:

```
"access_token":
"eb837eaf32459b711945a9d2259880119056e805ff0d2f36421cc171f94e58fef349f0005d217e36889b62271d9f00fc7cb5b1f
e5a86546d9dee8e22bca39b8f90d6cb61dae7fc383447823a09e380feeefba5bea0c994408470a49db0df32ddc2b0cca9381519e
60a63daae9b87ebfe9400b0c4af818b7f7d6c32e21465909efc8aa02804808f23ff96ac342b3b1c35230ac8858dbcb7979995d70
44c7b9cb05945c91b63a938703641e0fded339fb4c22e2383743a94a30c41892804193744e0c0f020909f9579555bf691b240faf
b558f76877fe0cb88ecbb3266fedbc7c541129270f67784d11ed658998b536841e0fdfc50a9ad056d2cabf717cb13326e4f620a6
ce172d8da4701b820c5ffe23223e7fb5725b244a1dd45538a0c7ca09a643759aaa2d8585a28689cae968ab3328351e3c38a8b199
040067ca5837169ce62a88282d1c8551d762fbdff77727cb51dd62213cef58dfb88e304abbc48063246b7e9f39650a0ac86f6c72
973b702b79faf34b68afb9412c9e0e56b104e12bf1ea3764faeac258fe1c3e896da412607a71ad8b4224efcfa0eafbc15f5e7af5
b8baa41163c220419c7249e9652ca7b5692b42cbe4c7d88c18d77440ec350582f51880e7354eed76ebd8ce760b27a6ca5808c6fc
51ef843ee5d98e5bbafaeb4096301853fa5fdc876275defd3ecabd0eb656c7cd2441e523e0c5468a1f261fb44"
```

2. Get your Platform UI access token by running the following cURL command:

```
curl -k X GET 'https://<hostname>/v1/preauth/validateAuth' \
-H 'username: admin' \
-H 'iam-token: <iam-token>'
```

   - `<hostname>` - The IBM Cloud Pak platform (`cpd`) route hostname.
   - `<username>` - Your username.
   - `<iam-token>` - The IAM access token that you obtained from the response in the previous step.

   The API response can resemble the following sample:

```
{
    "_messageCode_": "200",
    "message": "Success",
    "token": "<encoded-token-string>"
}
```

   - `<encoded-token-string>` - Your Platform UI access token.

Get authentication by using Platform UI API Key

1. Generate your platform API (Zen) key:

   1. Log in to the IBM Cloud Pak Automation console to authenticate your user credentials.

      For more information, such as for how to obtain the URL route for accessing the Automation console, see Accessing the Automation console.

   2. From the toolbar, click your avatar.
   3. Click **Profile and settings**.
   4. Click **API key** > **Generate new key**.
   5. Click **Generate**.
   6. Click **Copy** and save your key somewhere safe. You cannot recover this key if you lose it.

   If you lose your API key, repeat the preceding steps to generate a new API key. Your old API key becomes invalid, and any applications or scripts cannot authenticate to the platform until you provide your new API key. For more information, see Generating API keys for authentication.

2. Generate a base64 format of your Platform UI API Key:

```
ZENAPIKEY=$(echo "<username>:<API_key>" | base64 -w 0)
```

   - `<username>` - Your username.
   - `<API_key>` - The API key that you generated in the previous step.

3. The output is an encoded value. See the following example.

```
YWRtaW46MUFKblRIYngxNFJUVnBoR0FTdW0wcDJEOFl6UXFyVllmU2tmeW5qYgo=
```

   This encoded (base64 format) Platform UI API Key is used as the value for `<ZenApiKey>`.

4. Construct your Authorization header with the Platform UI API Key.

```
--header "Authorization: ZenApiKey <ZenApiKey>"
```

- `<ZenApiKey>` - The Platform UI API key that you get in the previous step.

5. Run the following cURL command:

```
curl "https://<Endpoint URL>" --header 'Content-Type: application/json' --header "Authorization:
ZenApiKey <ZenApiKey>" --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255' --insecure
```

- `<Endpoint URL>` - The Endpoint URL as specified on the API page of the service that you want to use, for example, you can find the Endpoint URL for metrics here.

Next steps

Access IBM Cloud Pak for AIOps API with your Platform UI authentication token or API keys. Include the Platform UI access token in the authorization header of your API calls.

For more information on the available API and related Swagger documentation, see APIs.

Enabling metrics monitoring in API gateway Kong

The page describes how to configure metrics monitoring in API gateway Kong based on Red Hat® OpenShift® Container Platform user-workload monitoring.

Prerequisites

1. Make sure that Kong is installed and all pods are running.

```
oc get pod -A | grep kong
```

Sample output:

```
NAMESPACE           NAME                                      READY   STATUS      RESTARTS   AGE
cp4aiops            gateway-kong-6774c76cd4-czg5v             2/2     Running     0          20d
cp4aiops            ibm-kong-operator-5b94c5b57b-8vbz2        1/1     Running     0          6d19h
cp4aiops            gateway-kong-config-svc-1615536000-kgjsf  0/1     Completed   0          6m36s
```

2. Make sure that the **ODLM** pod is running.

```
oc get pod -A | grep operand-deployment-lifecycle-manager
```

Sample output:

```
NAMESPACE              NAME                                                READY   STATUS      RESTARTS   AGE
ibm-common-services    operand-deployment-lifecycle-manager-8687fd5d45-zprqj   1/1     Running     0          97m
```

Deploy Kong monitoring

1. Obtain the `installation` instance name under the installation project (namespace) of IBM Cloud Pak® for AIOps.

```
oc get installation -A
```

As an example, the following sample output uses the name `cp4aiops` for the installation project (namespace) and `ibm-cp-aiops` for the instance name.

```
NAMESPACE   NAME           PHASE     LICENSE    STORAGECLASS    STORAGECLASSLARGEBLOCK
cp4aiops    ibm-cp-aiops   Running   Accepted   rook-cephfs     rook-cephfs
```

2. Edit the `installation` instance by running the following command:

```
oc edit installation <inst-name> -n <its-namespace>
```

3. Find the **config** section with name **aiopsFoundation** under **spec.pakModules**, for example:

```
spec:
  ...
  pakModules:
  - config:
    ...
    enabled: true
    name: aiopsFoundation
  - config:
```

4. If **aiopsFoundation** is not in **spec.pakModules.config**(see the following example), modify it to the example as shown in step 3.

```
spec:
  ...
  pakModules:
  - enabled: true
    name: aiopsFoundation
```

5. Under **spec.pakModules.config** with name **aiopsFoundation**, if **ibm-management-kong** does not exist, add **kong** section with metrics enabled as the following example:

```
- name: ibm-management-kong
  spec:
    kong:
      metrics:
        enabled: true
```

If **ibm-management-kong** exists, add the **metrics** section under **ibm-management-kong.spec.kong**:

```
metrics:
  enabled: true
```

Alternatively, you can run the following command to complete the preceding steps 2-5:

```
oc get install <inst-name> -n <its-namespace> -o yaml |  tr '\n' '\r' | sed -e 's/- enabled: true\r
name: aiopsFoundation/- config:\r    - name: ibm-management-kong\r      spec:\r        kong:\r
metrics:\r            enabled: true\r    enabled: true\r    name: aiopsFoundation/'  | tr '\r' '\n' | oc
apply -f -
```

After you complete your edits, the installation instance can look like the following sample:

```
oc get installation ibm-management -n cp4aiops -o yaml
```

Sample output:

```
apiVersion: orchestrator.aiops.ibm.com/v1alpha1
kind: Installation
metadata:
  generation: 4
  name: ibm-cp-aiops
  namespace: cp4aiops
spec:
  license:
    accept: true
  ...
  pakModules:
  ...
  - config:
    ...
    - name: ibm-management-kong
      spec:
        kong:
          metrics:
            enabled: true
    enabled: true
    name: aiopsFoundation
  - config:
     ...
```

6. Wait about 5 minutes to make sure that the derived **service**, **servicemonitor**,**MonitoringDashboard**, and **KongClusterPlugin** are created and running:

Run the following command:

```
oc get -A kcp
```

Sample output:

```
NAME          PLUGIN-TYPE   AGE
prometheus    prometheus    2m
```

Run the following command:

```
oc get servicemonitor -A | grep kong-metrics
```

Sample output:

```
cp4aiops                kong-metrics                    2m
```

Run the following command:

```
oc get MonitoringDashboard -A | grep kong-monitoring
```

Sample output:

```
ibm-common-services                kong-monitoring              true       2m
```

Check Kong monitoring dashboard in Grafana

1. In the Red Hat OpenShift Container Platform console, go to **Networking > Routes**, find the **cp-console** Route under `ibm-common-services` project (namespace), and click the url to log in to the IBM Cloud Pak Administration panel, for example, `https://cp-console.apps.xxx.com`, where `xxx` is the hostname in the environment.
2. Click **Monitoring** to go to Grafana UI, refresh the page if **Monitoring** is not shown.
3. Open dashboard dialog and click **API Management Monitoring** to view metrics monitoring dashboards.

Enabling logging in API gateway Kong

You can configure logging in the API gateway Kong that is installed in Red Hat® OpenShift® Container Platform.

Prerequisites

- Make sure that Kong is installed and all pods are running. To verify that the Kong pods are running, run the following command:

  ```
  oc get pod -A | grep kong
  ```

  Sample output:

  ```
  NAMESPACE              NAME                                READY   STATUS          RESTARTS   AGE
  kube-system            gateway-kong-6774c76cd4-czg5v       2/2     Running         0          20d
  kube-system            ibm-kong-operator-5b94c5b57b-8vbz2  1/1     Running         0          6d19h
  ```

- Make sure that Red Hat OpenShift Container Platform EFK stack for logging is installed. If not, follow instructions in Installing cluster logging to install. After installation, do the following steps to verify:

  1. Make sure that all `elasticsearch`, `fluent`, and `kibana` pods in project (namespace) `openshift-logging` run well.
  2. Make sure that kibana route is created. In the Red Hat OpenShift Container Platform web console, go to **Networking > Routes**, find the kibana Route under `openshift-logging` project (namespace).

Deploy Kong logging

1. Obtain the `installation` instance name under the installation project (namespace) of IBM Cloud Pak® for AIOps.

   ```
   oc get installation -A
   ```

   As an example, the following sample output uses the name `cp4aiops` for the installation project (namespace) and `ibm-cp-aiops` for the instance name.

   ```
   NAMESPACE      NAME                 LICENSE    STORAGECLASS
   cp4aiops       ibm-cp-aiops         Accepted   gp2
   ```

2. Edit the `installation` instance by running the following command:

   ```
   oc edit installation <inst-name> -n <its-namespace>
   ```

3. Find the **config** section with name **aiopsFoundation** under **spec.pakModules**, for example:

   ```
   spec:
     ...
     pakModules:
     - config:
       ...
       enabled: true
       name: aiopsFoundation
     - config:
   ```

4. If **aiopsFoundation** is not in **spec.pakModules.config**(see the following example), modify it to the example as shown in step 3.

   ```
   spec:
     ...
     pakModules:
   ```

```
  - enabled: true
    name: aiopsFoundation
```

5. Under **spec.pakModules.config** with name **aiopsFoundation**, if **ibm-management-kong** does not exist, add **kong** section with logging enabled as the following example:

```
- enable: true
   name: ibm-management-kong
   spec:
     kong:
       logging:
         enabled: true
```

If **ibm-management-kong** exists, add the **logging** section under `ibm-management-kong.spec.kong`:

```
logging:
  enabled: true
```

After you complete your edits, the installation instance can look like the following sample:

```
oc get installation ibm-management -n cp4aiops -o yaml
```

Sample output:

```
apiVersion: orchestrator.aiops.ibm.com/v1alpha1
kind: Installation
metadata:
  generation: 4
  name: ibm-cp-aiops
  namespace: cp4aiops
spec:
  imagePullSecret: ibm-cp-aiops-orchestrator
  license:
    accept: true
  pakModules:
  - enabled: false
    name: aiManager
  - config:
    - enabled: false
      name: ibm-watson-aiops-ui-operator
    - enabled: false
      name: abp-operator-lead
    - enable: true
      name: ibm-management-kong
      spec:
        kong:
          logging:
            enabled: true
    enabled: true
    name: aiopsFoundation
  - config:
    - enabled: false
      name: noi-operator
      spec:
        noi:
          clusterDomain: ""
    enabled: false
    name: applicationManager
  ...
```

6. Wait about 5 minutes to make sure that **svc**, **ds**, and **cm** are patched, and **KongClusterPlugin** is created:

    1. Check **cm** and **fluentd** for the included source and **<match kong>** after **</system>**:

```
oc get cm fluentd -n openshift-logging -o yaml  | grep "<system>" -A 19
  fluent.conf: |+
    ## CLO GENERATED CONFIGURATION ###
    # This file is a copy of the fluentd configuration entrypoint
    # which should normally be supplied in a configmap.

    <system>
      @log_level "#{ENV['LOG_LEVEL'] || 'warn'}"
    </system>
    <source>
      @type udp
      tag kong
      format json
    </source>
    <match kong>
      @type elasticsearch
      host elasticsearch.openshift-logging.svc.cluster.local
```

```
      port 9200
      verify_es_version_at_startup false
      scheme https
      ssl_version TLSv1_2
      index_name kong-000001
      client_key '/var/run/ocp-collector/secrets/fluentd/tls.key'
      client_cert '/var/run/ocp-collector/secrets/fluentd/tls.crt'
      ca_file '/var/run/ocp-collector/secrets/fluentd/ca-bundle.crt'
    </match>
```

2. Check **svc** and **fluentd** for **udp** port added:

```
oc get svc fluentd -n openshift-logging -o jsonpath='{.spec.ports}'
[{"name":"udp","port":5160,"protocol":"UDP","targetPort":5160},
{"name":"metrics","port":24231,"protocol":"TCP","targetPort":"metrics"}]
```

3. Check **ds/fluentd** for **udp** port added:

```
oc get  ds/fluentd -n openshift-logging -o jsonpath='{.spec.template.spec.containers[0].ports}'
[{"containerPort":5160,"name":"udp","protocol":"UDP"},
{"containerPort":24231,"name":"metrics","protocol":"TCP"}]
```

4. Check **KongClusterPlugin**:

```
oc get -A  kcp

NAME          PLUGIN-TYPE   AGE
udp-log       udp-log       2m
```

Import Kong logging dashboard in kibana

In the Red Hat OpenShift Container Platform web console, go to **Networking > Routes**, find the kibana Route under **openshift-logging** project (namespace), and click the url to log in to Kibana, for example, **https://kibana-openshift-logging.apps.xxx.com**, in which **xxx** is the hostname in the environment. Then, import logging dashboards by selecting a JSON file. See the following JSON file for example.

```
[
  {
    "_id": "7ff715b0-33a2-11eb-8048-4501952b91a3",
    "_type": "dashboard",
    "_source": {
      "title": "API Dashboard",
      "hits": 0,
      "description": "",
      "panelsJSON": "[{\"gridData\":
{\"x\":0,\"y\":0,\"w\":24,\"h\":15,\"i\":\"1\"},\"version\":\"6.8.1\",\"panelIndex\":\"1\",\"type\":\"visuali
zation\",\"id\":\"a60b06a0-339b-11eb-8048-4501952b91a3\",\"embeddableConfig\":{}},{\"gridData\":
{\"x\":24,\"y\":0,\"w\":24,\"h\":15,\"i\":\"2\"},\"version\":\"6.8.1\",\"panelIndex\":\"2\",\"type\":\"visual
ization\",\"id\":\"5de48cf0-33a2-11eb-8048-4501952b91a3\",\"embeddableConfig\":{}},{\"gridData\":
{\"x\":0,\"y\":15,\"w\":24,\"h\":15,\"i\":\"3\"},\"version\":\"6.8.1\",\"panelIndex\":\"3\",\"type\":\"visual
ization\",\"id\":\"4a788b50-33a0-11eb-8048-4501952b91a3\",\"embeddableConfig\":{}},{\"gridData\":
{\"w\":24,\"h\":15,\"x\":24,\"y\":15,\"i\":\"4\"},\"version\":\"6.8.1\",\"panelIndex\":\"4\",\"type\":\"visua
lization\",\"id\":\"208b1870-33a1-11eb-8048-4501952b91a3\",\"embeddableConfig\":{}}]",
      "optionsJSON": "{\"darkTheme\":false,\"useMargins\":true,\"hidePanelTitles\":false}",
      "version": 1,
      "timeRestore": false,
      "kibanaSavedObjectMeta": {
        "searchSourceJSON": "{\"query\":{\"query\":\"\",\"language\":\"lucene\"},\"filter\":[]}"
      }
    }
  },
  {
    "_id": "977ad660-338c-11eb-8048-4501952b91a3",
    "_type": "index-pattern",
    "_source": {
      "title": "kong-*",
      "fields": "
[{\"name\":\"_id\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":tr
ue,\"readFromDocValues\":false},
{\"name\":\"_index\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":
true,\"readFromDocValues\":false},
{\"name\":\"_score\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":false,\"aggregatable\"
:false,\"readFromDocValues\":false},
{\"name\":\"_source\",\"type\":\"_source\",\"count\":0,\"scripted\":false,\"searchable\":false,\"aggregatable
\":false,\"readFromDocValues\":false},
{\"name\":\"_type\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":t
rue,\"readFromDocValues\":false},
{\"name\":\"client_ip\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable
\":false,\"readFromDocValues\":false},
{\"name\":\"client_ip.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggr
egatable\":true,\"readFromDocValues\":true},
```

{\"name\":\"latencies.kong\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"latencies.proxy\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"latencies.request\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"request.headers.accept\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"request.headers.accept.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"request.headers.host\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"request.headers.host.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},{\"name\":\"request.headers.user-agent\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},{\"name\":\"request.headers.user-agent.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"request.method\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"request.method.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"request.size\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"request.size.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"request.uri\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"request.uri.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"request.url\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"request.url.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"response.headers.connection\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"response.headers.connection.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},{\"name\":\"response.headers.content-length\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},{\"name\":\"response.headers.content-length.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},{\"name\":\"response.headers.content-type\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},{\"name\":\"response.headers.content-type.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"response.headers.server\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"response.headers.server.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},{\"name\":\"response.headers.x-kong-response-latency\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},{\"name\":\"response.headers.x-kong-response-latency.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"response.size\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"response.size.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"response.status\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.created_at\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.https_redirect_status_code\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.id\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"route.id.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.name\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"route.name.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.path_handling\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"route.path_handling.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.paths\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":false,\"readFromDocValues\":false},
{\"name\":\"route.paths.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.preserve_host\",\"type\":\"boolean\",\"count\":0,\"scripted\":false,\"searchable\":true,\"a

```
ggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.protocols\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggreg
atable\":false,\"readFromDocValues\":false},
{\"name\":\"route.protocols.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,
\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.regex_priority\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"a
ggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.service.id\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggre
gatable\":false,\"readFromDocValues\":false},
{\"name\":\"route.service.id.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true
,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.strip_path\",\"type\":\"boolean\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggr
egatable\":true,\"readFromDocValues\":true},
{\"name\":\"route.updated_at\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggre
gatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.connect_timeout\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,
\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.created_at\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"agg
regatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.host\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregata
ble\":false,\"readFromDocValues\":false},
{\"name\":\"service.host.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"a
ggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.id\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatabl
e\":false,\"readFromDocValues\":false},
{\"name\":\"service.id.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"agg
regatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.name\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregata
ble\":false,\"readFromDocValues\":false},
{\"name\":\"service.name.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"a
ggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.path\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregata
ble\":false,\"readFromDocValues\":false},
{\"name\":\"service.path.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"a
ggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.port\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregata
ble\":true,\"readFromDocValues\":true},
{\"name\":\"service.protocol\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggre
gatable\":false,\"readFromDocValues\":false},
{\"name\":\"service.protocol.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true
,\"aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.read_timeout\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"a
ggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.retries\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggreg
atable\":true,\"readFromDocValues\":true},
{\"name\":\"service.updated_at\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"agg
regatable\":true,\"readFromDocValues\":true},
{\"name\":\"service.write_timeout\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"
aggregatable\":true,\"readFromDocValues\":true},
{\"name\":\"started_at\",\"type\":\"number\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregatabl
e\":true,\"readFromDocValues\":true},
{\"name\":\"upstream_uri\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"aggregata
ble\":false,\"readFromDocValues\":false},
{\"name\":\"upstream_uri.keyword\",\"type\":\"string\",\"count\":0,\"scripted\":false,\"searchable\":true,\"a
ggregatable\":true,\"readFromDocValues\":true}]"
    }
  },
  {
    "_id": "a60b06a0-339b-11eb-8048-4501952b91a3",
    "_type": "visualization",
    "_source": {
      "title": "API Performance",
      "visState": "{\"title\":\"API Performance\",\"type\":\"table\",\"params\":
{\"perPage\":10,\"showPartialRows\":false,\"showMetricsAtAllLevels\":false,\"sort\":
{\"columnIndex\":null,\"direction\":null},\"showTotal\":false,\"totalFunc\":\"sum\"},\"aggs\":
[{\"id\":\"1\",\"enabled\":true,\"type\":\"count\",\"schema\":\"metric\",\"params\":{}},
{\"id\":\"2\",\"enabled\":true,\"type\":\"terms\",\"schema\":\"bucket\",\"params\":
{\"field\":\"route.paths.keyword\",\"size\":5,\"order\":\"desc\",\"orderBy\":\"1\",\"otherBucket\":false,\"ot
herBucketLabel\":\"Other\",\"missingBucket\":false,\"missingBucketLabel\":\"Missing\",\"customLabel\":\"Route
\"}},{\"id\":\"3\",\"enabled\":true,\"type\":\"terms\",\"schema\":\"bucket\",\"params\":
{\"field\":\"response.status\",\"size\":5,\"order\":\"desc\",\"orderBy\":\"1\",\"otherBucket\":false,\"otherB
ucketLabel\":\"Other\",\"missingBucket\":false,\"missingBucketLabel\":\"Missing\",\"customLabel\":\"Response
status\"}},{\"id\":\"4\",\"enabled\":true,\"type\":\"avg\",\"schema\":\"metric\",\"params\":
{\"field\":\"latencies.request\",\"customLabel\":\"Latency (sec)\"}}]}",
      "uiStateJSON": "{\"vis\":{\"params\":{\"sort\":{\"columnIndex\":null,\"direction\":null}}}}",
      "description": "",
      "version": 1,
      "kibanaSavedObjectMeta": {
        "searchSourceJSON": "{\"index\":\"977ad660-338c-11eb-8048-4501952b91a3\",\"query\":
{\"query\":\"\",\"language\":\"lucene\"},\"filter\":[]}"
      }
```

```
    },
    "_migrationVersion": {
      "visualization": "6.7.2"
    }
  },
  {
    "_id": "4a788b50-33a0-11eb-8048-4501952b91a3",
    "_type": "visualization",
    "_source": {
      "title": "Busiest services",
      "visState": "{\"title\":\"Busiest services\",\"type\":\"histogram\",\"params\":
{\"type\":\"histogram\",\"grid\":{\"categoryLines\":false,\"style\":{\"color\":\"#eee\"}},\"categoryAxes\":
[{\"id\":\"CategoryAxis-1\",\"type\":\"category\",\"position\":\"bottom\",\"show\":true,\"style\":
{},\"scale\":{\"type\":\"linear\"},\"labels\":{\"show\":true,\"truncate\":100},\"title\":{}}],\"valueAxes\":
[{\"id\":\"ValueAxis-1\",\"name\":\"LeftAxis-
1\",\"type\":\"value\",\"position\":\"left\",\"show\":true,\"style\":{},\"scale\":
{\"type\":\"linear\",\"mode\":\"normal\"},\"labels\":
{\"show\":true,\"rotate\":0,\"filter\":false,\"truncate\":100},\"title\":
{\"text\":\"Count\"}}],\"seriesParams\":
[{\"show\":\"true\",\"type\":\"histogram\",\"mode\":\"stacked\",\"data\":
{\"label\":\"Count\",\"id\":\"1\"},\"valueAxis\":\"ValueAxis-
1\",\"drawLinesBetweenPoints\":true,\"showCircles\":true}],\"addTooltip\":true,\"addLegend\":true,\"legendPos
ition\":\"right\",\"times\":[],\"addTimeMarker\":false},\"aggs\":
[{\"id\":\"1\",\"enabled\":true,\"type\":\"count\",\"schema\":\"metric\",\"params\":{}},
{\"id\":\"2\",\"enabled\":true,\"type\":\"terms\",\"schema\":\"segment\",\"params\":
{\"field\":\"service.name.keyword\",\"size\":5,\"order\":\"desc\",\"orderBy\":\"1\",\"otherBucket\":false,\"o
therBucketLabel\":\"Other\",\"missingBucket\":false,\"missingBucketLabel\":\"Missing\",\"customLabel\":\"Serv
ice\"}},{\"id\":\"3\",\"enabled\":true,\"type\":\"terms\",\"schema\":\"group\",\"params\":
{\"field\":\"response.status\",\"size\":5,\"order\":\"desc\",\"orderBy\":\"1\",\"otherBucket\":false,\"otherB
ucketLabel\":\"Other\",\"missingBucket\":false,\"missingBucketLabel\":\"Missing\",\"customLabel\":\"Response
code\"}}]}",
      "uiStateJSON": "{}",
      "description": "",
      "version": 1,
      "kibanaSavedObjectMeta": {
        "searchSourceJSON": "{\"index\":\"977ad660-338c-11eb-8048-4501952b91a3\",\"query\":
{\"query\":\"\",\"language\":\"lucene\"},\"filter\":[]}"
      }
    },
    "_migrationVersion": {
      "visualization": "6.7.2"
    }
  },
  {
    "_id": "208b1870-33a1-11eb-8048-4501952b91a3",
    "_type": "visualization",
    "_source": {
      "title": "Slowest services",
      "visState": "{\"title\":\"Slowest services\",\"type\":\"horizontal_bar\",\"params\":
{\"type\":\"histogram\",\"grid\":{\"categoryLines\":false,\"style\":{\"color\":\"#eee\"}},\"categoryAxes\":
[{\"id\":\"CategoryAxis-1\",\"type\":\"category\",\"position\":\"left\",\"show\":true,\"style\":{},\"scale\":
{\"type\":\"linear\"},\"labels\":{\"show\":true,\"rotate\":0,\"filter\":false,\"truncate\":200},\"title\":
{}}],\"valueAxes\":[{\"id\":\"ValueAxis-1\",\"name\":\"LeftAxis-
1\",\"type\":\"value\",\"position\":\"bottom\",\"show\":true,\"style\":{},\"scale\":
{\"type\":\"linear\",\"mode\":\"normal\"},\"labels\":
{\"show\":true,\"rotate\":75,\"filter\":true,\"truncate\":100},\"title\":
{\"text\":\"Latency\"}}],\"seriesParams\":
[{\"show\":true,\"type\":\"histogram\",\"mode\":\"normal\",\"data\":
{\"label\":\"Latency\",\"id\":\"1\"},\"valueAxis\":\"ValueAxis-
1\",\"drawLinesBetweenPoints\":true,\"showCircles\":true}],\"addTooltip\":true,\"addLegend\":true,\"legendPos
ition\":\"right\",\"times\":[],\"addTimeMarker\":false},\"aggs\":
[{\"id\":\"1\",\"enabled\":true,\"type\":\"avg\",\"schema\":\"metric\",\"params\":
{\"field\":\"latencies.request\",\"customLabel\":\"Latency\"}},
{\"id\":\"2\",\"enabled\":true,\"type\":\"terms\",\"schema\":\"segment\",\"params\":
{\"field\":\"service.name.keyword\",\"size\":5,\"order\":\"desc\",\"orderBy\":\"1\",\"otherBucket\":false,\"o
therBucketLabel\":\"Other\",\"missingBucket\":false,\"missingBucketLabel\":\"Missing\",\"customLabel\":\"Serv
ice\"}}]}",
      "uiStateJSON": "{}",
      "description": "",
      "version": 1,
      "kibanaSavedObjectMeta": {
        "searchSourceJSON": "{\"index\":\"977ad660-338c-11eb-8048-4501952b91a3\",\"query\":
{\"query\":\"\",\"language\":\"lucene\"},\"filter\":[]}"
      }
    },
    "_migrationVersion": {
      "visualization": "6.7.2"
    }
  },
  {
    "_id": "5de48cf0-33a2-11eb-8048-4501952b91a3",
```

```
    "_type": "visualization",
    "_source": {
      "title": "Inter-service communication",
      "visState": "{\"title\":\"Inter-service communication\",\"type\":\"heatmap\",\"params\":
{\"type\":\"heatmap\",\"addTooltip\":true,\"addLegend\":true,\"enableHover\":false,\"legendPosition\":\"right
\",\"times\":[],\"colorsNumber\":4,\"colorSchema\":\"Greens\",\"setColorRange\":false,\"colorsRange\":
[],\"invertColors\":false,\"percentageMode\":false,\"valueAxes\":[{\"show\":false,\"id\":\"ValueAxis-
1\",\"type\":\"value\",\"scale\":{\"type\":\"linear\",\"defaultYExtents\":false},\"labels\":
{\"show\":false,\"rotate\":0,\"overwriteColor\":false,\"color\":\"#555\"}}]},\"aggs\":
[{\"id\":\"1\",\"enabled\":true,\"type\":\"count\",\"schema\":\"metric\",\"params\":{}},
{\"id\":\"2\",\"enabled\":true,\"type\":\"terms\",\"schema\":\"group\",\"params\":
{\"field\":\"service.name.keyword\",\"size\":5,\"order\":\"desc\",\"orderBy\":\"1\",\"otherBucket\":false,\"o
therBucketLabel\":\"Other\",\"missingBucket\":false,\"missingBucketLabel\":\"Missing\",\"customLabel\":\"Serv
ice\"}},{\"id\":\"3\",\"enabled\":true,\"type\":\"terms\",\"schema\":\"segment\",\"params\":
{\"field\":\"client_ip.keyword\",\"size\":5,\"order\":\"desc\",\"orderBy\":\"1\",\"otherBucket\":false,\"othe
rBucketLabel\":\"Other\",\"missingBucket\":false,\"missingBucketLabel\":\"Missing\",\"customLabel\":\"Client
IP\"}}]}",
      "uiStateJSON": "{\"vis\":{\"defaultColors\":{\"0 - 0.5\":\"rgb(247,252,245)\",\"0.5 -
1\":\"rgb(199,233,192)\",\"1 - 1.5\":\"rgb(116,196,118)\",\"1.5 - 2\":\"rgb(35,139,69)\"}}}",
      "description": "",
      "version": 1,
      "kibanaSavedObjectMeta": {
        "searchSourceJSON": "{\"index\":\"977ad660-338c-11eb-8048-4501952b91a3\",\"query\":
{\"query\":\"\",\"language\":\"lucene\"},\"filter\":[]}"
      }
    },
    "_migrationVersion": {
      "visualization": "6.7.2"
    }
  }
]
```

Security test

Log in to the Red Hat OpenShift Container Platform web console as a different user. Go to the Kibana and try to check the logging dashboard. The imported logging dashboard is not visible for a different user.

Platform UI APIs

Learn more about the Platform UI APIs that are available for you to use to manage users and the personalization of the console.

User management APIs

Create and manage users and access to the console with the user management API.

Custom cards API

Create custom cards to display information on the console home page with the Custom cards REST API.

Generating API keys for authentication

With an API key, you can automatically authenticate to the platform or to a specific instance of a service from a script or application. Your API keys are associated with your credentials and are specific to you. With API keys, you can authenticate without entering your username and password.

You can generate:

- A platform API key

Your platform API key enables scripts and applications to access everything that you would typically be able to access when you log in to the console.

Before you begin

Before you generate an API key, consider the following factors:

What level of access is required?

As a security best practice, it is recommended that you give the least amount of access necessary to the script or application. For example, if a script needs to interact only with a specific service instance, generate an API key for that service instance.

Important: Be sure to store your API keys somewhere safe. If you lose an API key, you'll need to create a new key and update any scripts or applications where the key is used.

Platform API key

With a platform API key, you can access everything that you would typically be able to access when you log in to the console.

To generate a platform API key:

1. Log in to the console.
2. From the toolbar, click your avatar.
3. Click **Profile and settings**.
4. Click **API key** > **Generate new key**.
5. Click **Generate**.
6. Click **Copy** and save your key somewhere safe. You cannot recover this key if you lose it.

If you lose your API key, repeat the preceding steps to generate a new API key. Your old API key becomes invalid, and any applications or scripts cannot authenticate to the platform until you provide your new API key.

If you believe that your API key is compromised:

1. Log in to the console.
2. From the toolbar, click your avatar.
3. Click **Profile and settings**.
4. Click **API key** > **Revoke current key**.
5. Click **Revoke**.

Any applications or scripts that use the key won't be able to authenticate to the platform.

IAM APIs

Identity and Access management (IAM) APIs.

- Preparing to run API commands
- User management and authentication API
- Single sign-on APIs
- Service health and version check APIs

Preparing to run component or management API commands

Before you run component API commands, retrieve the authentication token and download the CA certificate for your cluster.

Before you begin

You can use the IBM Cloud Pak CLI (cloudctl) or cURL commands to retrieve your token. For more information about installing the cloudctl tool, see Install the IBM Cloud CLI (cloudctl).

Procedure

1. Retrieve the authentication tokens. You can use cloudctl or run curl commands.

   - To use cloudctl, run the following command:

     ```
     cloudctl tokens
     ```

     The access token and ID token display:

     ```
     Access token:  Bearer
     jHVcaGjSuWEXGUPeH8WVnoqUyex5kJDlC14DnZMQN2WseErIoVWDW5cY1Ikvoqdhxkou4bvHYJM77U2FZf2aKbo3h2FeH1OJpaE
     wj8rpGiDLusaXc54rJbnefbjAv4OECvk2tg9gIeJAx6DZlmlknDXB6zM0bkGtqayP94L8gSXtzPoW6miuoR8dFD5Du63OjVBVZl
     jw5ajOWBtvYzR1ttamH4SRMflEVWC3AxcBhtkadMCAsdWFi1KRGUoVM5k0TkuPFP7AAX8vryEpakeVqBQfbR2m0OlRBTRbkNJoa
     cPp7AesyGLBjPVElQs01FCz3xhMNHkQJAFT0T0xhbhDj3DgenAjJd5eSNWxdTLSfhOaQC1fx6BLq5Z0e8S0Uhp2NSWqNVkU49fH
     uTHW2UwxG84nvwpz4ZxYJaP3m6DZv5lTwKcbQkB3w9cLr8pgUaC94IZ9gJhYvprMR1NKz4etFgNPmhA4T1NS8NVVgMHBYOptv1q
     AdKSNzZ2l6V23h2BquFpVQ2MqE6Lcx0N0j1ZoRVuYo1qTrIkOpEBNbn94b3PHNJMVO2v1NqV85G6uNgPdvv85eneHItsIfUc5yX
     MeYLXZ017trlhxj43lLVsEyUaM7S5dfykayqVgsaJ9faHYz6F14oRKJCbwhg7y4ybxxxR5KtCKihXgf5QJJXYmsrimudD8KXMVs
     ```

```
QKEyHFETzVR7eNMRZvcohFKBPwZdIntkcpLygMmZK4Gaz6pD3t4PTkVXKAWQsPTepJ57FfP9kNkPU6BDxf9X7skBrsRFD3ldCdF
UMi0lxOmkBakmXHLODRmmZl73bKuIKFT7lZAXY0Rn3d5NzXcyzVODRNXnMyRW9c2NLvhYQe0ltdR8dIDtRbsh1AQEC5P1eI9XGh
BLcAqhU0oBe7k5rMzQXL44TtYMo9NccsgMjNsfToyCqezocgfmLYjqZOUnRbBG7ymn5FJECzOd8X1KagaXqzlSmDtD7cRzZms3i
A8FE85cWaK
ID token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiN2p2MmFjeGtucHVpcXo1MXRna2giLCJyZWFsbU5hbWUiOi
JjdXN0b21SZWFsbSIsInVuaXF1ZVNlY3VyaXR5TmFtZSI6ImFkbWluIiwiaXNzIjoiaHR0cHM6Ly9teWNsdXN0ZXIuaWNwOjk0N
DMvb2lkYy9lbmRwb2ludC9PUCIsImF1ZCI6ImY4YjVjZGE1YTgzZjg4NjZhOTIxMTQ2MGU5YTk4YzQ4IiwiZXhwIjoxNTA4MjYw
ODc4LCJpYXQiOjE1MDgyNjA4NzgsInN1YiI6ImFkbWluIn0.IrLm1R9a4GBiTG0wYR1JhGqT4HSArN3gPHhPPTC4ZuS46LulRQC
Bksxh9I59uT4pYcqhd0qJ_xp9Ys1H8xLsq1zKSI0W2KAzuFkIbXQiK9Q6_Z3oQOHE8XMG7Xfb0R8B4TgbTjQ3XWkEkXsyeliXk0
l7mqlVIgTFbXx8nqcoFbXhmH7ZQukj731MQ0AyKKPpJktWtPCLpugtiTA0nkKUodncvHdSw43bmVQuGsQ_kRhhr8Ka8y_olYcBt
YUSAKqdwiGPu6O0Qk-57FCiUmX4W9pjLRAR9EmILY9RqJAsH5kE11kYHPTO2fu-B6omzw2eKxhjZYHMIPmxUciiBRB9Pw
```

These tokens are stored in the **/<user_folder>/.cloudctl/config.json** file while you are logged in to the CLI, where **<user_folder>** is the path to your user directory, such as **/Users/my_username** on macOS.

- To use curl, run the following command,

```
curl -k -H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8" -d
"grant_type=password&username=admin&password=admin&scope=openid"
https://<cluster_address>/idprovider/v1/auth/identitytoken
```

The command returns an **access_token**, **refresh_token**, and **id_token**, as shown in the following example:

```
{"access_token":"eb837eaf32459b711945a9d2259880119056e805ff0d2f36421cc171f94e58fef349f0005d217e3688
9b62271d9f00fc7cb5b1fe5a86546d9dee8e22bca39b8f90d6cb61dae7fc383447823a09e380feeefba5bea0c994408470a
49db0df32ddc2b0cca9381519e60a63daae9b87ebfe9400b0c4af818b7f7d6c32e21465909efc8aa02804808f23ff96ac34
2b3b1c35230ac8858dbcb7979995d7044c7b9cb05945c91b63a938703641e0fded339fb4c22e2383743a94a30c418928041
93744e0c0f020909f9579555bf691b240fafb558f76877fe0cb88ecbb3266fedbc7c541129270f67784d11ed658998b5368
41e0fdfc50a9ad056d2cabf717cb13326e4f620a6ce172d8da4701b820c5ffe23223e7fb5725b244a1dd45538a0c7ca09a6
43759aaa2d8585a28689cae968ab3328351e3c38a8b199040067ca5837169ce62a88282d1c8551d762fbdff77727cb51dd6
2213cef58dfb88e304abbc48063246b7e9f39650a0ac86f6c72973b702b79faf34b68afb9412c9e0e56b104e12bf1ea3764
faeac258fe1c3e896da412607a71ad8b4224efcfa0eafbc15f5e7af5b8baa41163c220419c7249e9652ca7b5692b42cbe4c
7d88c18d77440ec350582f51880e7354eed76ebd8ce760b27a6ca5808c6fc51ef843ee5d98e5bbafaeb4096301853fa5fdc
876275defd3ecabd0eb656c7cd2441e523e0c5468a1f261fb44","token_type":"Bearer","expires_in":43199,"scop
e":"openid","refresh_token":"6q4griAg9yCiGINQvF0Dp7N9hqXhcXZrAsqWWYgl6XQ80Uexsq","id_token":"eyJ0eX
AiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiYWRmZDc4MmEwOTc1ZTNmMzc2ZTkxZTI3YjJkNTYxZmQ0OTNiNTQz
MSIsInJlYWxtTmFtZSI6ImN1c3RvbVJlYWxtIiwidW5pcXVlU2VjdXJpdHlOYW1lIjoiYWRtaW4iLCJpc3MiOiJodHRwczovL21
5Y2x1c3Rlci5pY3A6OTQ0My9vaWRjL2VuZHBvaW50L09QIiwiYXVkIjoiMGQzYzA3MTc5OTYxYmEzMWEyODY5NDU0NDQwM2E0ND
YiLCJleHAiOjE1NTQ5MTQ2NTIsImlhdCI6MTU1NDg4NTg1Miwic3ViIjoiYWRtaW4iLCJ0ZWFtUm9sZU1hcHBpbmdzIjpbXX0.C
nT0qWECpJR9R16W-IOqrXjSJR8DelRsDUXcX6hy_I0DPQ7hU55Bhcq6UChEg3qiWWRbKwrFIxikXPjEjw2B9oziEd8U8AEO-
4LEaXOpc5Lk1shvyxBQFDDgyUwgyGb-erRbO_Sl1K4xotuTLg4nhoydwTXs7lZn7GC4UW8j1qkhlbFe5iLgKidCZsjyPo-
2GNYEQn0ufHH3KCR4DkHi6GX2RUxisNecwDzNl9P5JSyjlS-
r5QUZJ0b0DytKuY5HxpswpIFaO9U8JlYAFoOZ18eO_CzERHRQ_Ii1ePmagGAk-
eLJjmCNqY1zynfpEUuKlWUR5rVGHGzSbGA8J4CLvg"}
```

From the example, following is the **access token**:

```
"access_token":
"eb837eaf32459b711945a9d2259880119056e805ff0d2f36421cc171f94e58fef349f0005d217e36889b62271d9f00fc7c
b5b1fe5a86546d9dee8e22bca39b8f90d6cb61dae7fc383447823a09e380feeefba5bea0c994408470a49db0df32ddc2b0c
ca9381519e60a63daae9b87ebfe9400b0c4af818b7f7d6c32e21465909efc8aa02804808f23ff96ac342b3b1c35230ac885
8dbcb7979995d7044c7b9cb05945c91b63a938703641e0fded339fb4c22e2383743a94a30c41892804193744e0c0f020909
f9579555bf691b240fafb558f76877fe0cb88ecbb3266fedbc7c541129270f67784d11ed658998b536841e0fdfc50a9ad05
6d2cabf717cb13326e4f620a6ce172d8da4701b820c5ffe23223e7fb5725b244a1dd45538a0c7ca09a643759aaa2d8585a2
8689cae968ab3328351e3c38a8b199040067ca5837169ce62a88282d1c8551d762fbdff77727cb51dd62213cef58dfb88e3
04abbc48063246b7e9f39650a0ac86f6c72973b702b79faf34b68afb9412c9e0e56b104e12bf1ea3764faeac258fe1c3e89
6da412607a71ad8b4224efcfa0eafbc15f5e7af5b8baa41163c220419c7249e9652ca7b5692b42cbe4c7d88c18d77440ec3
50582f51880e7354eed76ebd8ce760b27a6ca5808c6fc51ef843ee5d98e5bbafaeb4096301853fa5fdc876275defd3ecabd
0eb656c7cd2441e523e0c5468a1f261fb44"
```

From the example, following is the **refresh token**:

```
"refresh_token":"6q4griAg9yCiGINQvF0Dp7N9hqXhcXZrAsqWWYgl6XQ80Uexsq"
```

From the example, following is the **ID token**:

```
"id_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiYWRmZDc4MmEwOTc1ZTNmMzc2ZTkxZTI3Yj
JkNTYxZmQ0OTNiNTQzMSIsInJlYWxtTmFtZSI6ImN1c3RvbVJlYWxtIiwidW5pcXVlU2VjdXJpdHlOYW1lIjoiYWRtaW4iLCJpc
3MiOiJodHRwczovL215Y2x1c3Rlci5pY3A6OTQ0My9vaWRjL2VuZHBvaW50L09QIiwiYXVkIjoiMGQzYzA3MTc5OTYxYmEzMWEy
ODY5NDU0NDQwM2E0NDYiLCJleHAiOjE1NTQ5MTQ2NTIsImlhdCI6MTU1NDg4NTg1Miwic3ViIjoiYWRtaW4iLCJ0ZWFtUm9sZU1
hcHBpbmdzIjpbXX0.CnT0qWECpJR9R16W-
IOqrXjSJR8DelRsDUXcX6hy_I0DPQ7hU55Bhcq6UChEg3qiWWRbKwrFIxikXPjEjw2B9oziEd8U8AEO-
4LEaXOpc5Lk1shvyxBQFDDgyUwgyGb-erRbO_Sl1K4xotuTLg4nhoydwTXs7lZn7GC4UW8j1qkhlbFe5iLgKidCZsjyPo-
2GNYEQn0ufHH3KCR4DkHi6GX2RUxisNecwDzNl9P5JSyjlS-
r5QUZJ0b0DytKuY5HxpswpIFaO9U8JlYAFoOZ18eO_CzERHRQ_Ii1ePmagGAk-
eLJjmCNqY1zynfpEUuKlWUR5rVGHGzSbGA8J4CLvg"
```

2. Store the authentication token in a variable. You can access IBM Cloud Pak® for AIOps APIs by specifying an authentication token in the request header. Run the following command, where **<ID token>** is the displayed ID token:

```
export ID_TOKEN=<ID token>
```

3. Store the access token in a variable. Include the full contents of the access token, including the Bearer value. For example, from the access token in the Curl command output in step 3, you must include the token value from **"eb837e** to **1fb44"**. You can access IBM Cloud Pak for AIOps user management APIs by specifying the access token in the request header. Run the following command, where **<Access token>** is the following displayed access token:

```
export ACCESS_TOKEN=<Access token>
```

4. Obtain a copy of the CA certificate for your cluster.

   - If you can access the boot node, the CA certificate file is **/<installation_directory>/cluster/cfc-certs/root-ca/ca.crt**.
   - To use cloudctl:
     - Ensure that you have logged in with cloudctl as required. This places the cluster's certificates into the cloudctl configuration directory.
     - Confirm that the authentication certificate is available. Run the following command, where **<user_folder>** is the path to your user home directory, such as **/Users/my_username** on macOS, and **<cluster>** is your cluster name. This file path is the **<certificate_path>** variable that you use in a later step:

     ```
     ``` {: codeblock}
     ls <user_folder>/.cloudctl/clusters/<cluster_name>
     ```
     The `ca.crt` file displays, as it is in the following output:

     ``` {: pre}
     ca.pem          cert.pem        key.pem         kube-config     kube-config.bat
     ```
     ```

User management and authentication APIs

The user management and authentication APIs manage users and teams.

To use these APIs, you must add an authorization header to your request. You need an access token to add to the authorization header. To obtain the access token, see Preparing to run component or management API commands.

User management and authentication API data

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Base path

idmgmt/identity/api/v1

Command output format

application/json

- Account management
- Directory management
- User group management
- User management

Account management APIs

APIs to manage accounts.

Base path: **`https://<cluster_address>/idmgmt/identity/api/v1/account`**

Get information about all accounts

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/account

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/account'
```

The response resembles the following code:

```
[{"id":"id-mycluster-account","name":"mycluster Account","description":"Description for mycluster Account"}]
```

Get information about an account

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/account/{id}

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/account/id-mycluster-account'
```

The response resembles the following code:

```
{"id":"id-mycluster-account","name":"mycluster Account","description":"Description for mycluster Account","url":"/identity/api/v1/account/id-mycluster-account"}
```

Directory management APIs

APIs to manage the LDAP directory.

Base path: **https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap**

Note:

- From IBM Cloud Pak foundational services version 3.23,**LDAP_TDS_NESTEDCONFIG** parameter is introduced in directory management API. The value of this parameter is used to decide whether recursive method or TDS/SDS recommended way is used to enable nested search. If the parameter is not provided, its default value will be considered as **false**. If this parameter is set as **true**, the TDS/SDS recommended logic will be used for nested search. If this parameter is set as **false**, the recursive method logic will be used for nested search. For more information, see Enabling LDAP Nested Search.
- From the foundational services version 3.22 and later, **LDAP_USER_SEARCHBASE_LIST** parameter is introduced in directory management API to enable custom search base for LDAP user entity through SCIM user APIs. It is an optional parameter. For more information, see Custom search base support for LDAP group and user entity in SCIM group and user APIs.
- From the foundational services version 3.21 and later, the following parameters are supported in the Directory management API:
  - **LDAP_PAGINGSEARCH**: It is an optional parameter. If the **LDAP_PAGINGSEARCH** parameter is set as "false", it indicates that the pagination option is disabled. Set this parameter to "true" to enable the pagination option.
  - **LDAP_PAGING_SIZE**: It is an optional parameter. It is a numeric field that defines the page size of the LDAP type. You can set the number as per your requirement. It is an optional field. If you do not provide the value to the **LDAP_PAGING_SIZE** parameter, the default page size of the LDAP server will be considered as a request in the API query. For example, when the LDAP type is Microsoft Active Directory (MSAD) and you do not provide the **LDAP_PAGING_SIZE** value in the API, then 1000 (default page size) value will be considered as a request in the API. Also, the **LDAP_PAGING_SIZE** value should not exceed the default server side page limit or the maximum configured value for the respective LDAP type. For more information, see SCIM pagination support for LDAP server.
  - **LDAP_GROUP_SEARCHBASE_LIST**: It is an optional parameter. You can use **LDAP_GROUP_SEARCHBASE_LIST** to enable the custom search base on LDAP group entity through SCIM group APIs.

For more information, see Custom search base support for LDAP group and user entity in SCIM group and user APIs.

Creating an LDAP connection

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/onboardDirectory

Command

POST

Command output format

application/json

Note:

- There is a limitation on liberty, that is, the **LDAP_USERFILTER** and **LDAP_GROUPFILTER** fields must contain an attribute with **=%v** value. For example, **(cn=%v)** for Cloud Pak and Cloud Pak for data login flow to work.
- If you are setting up multiple LDAP connections, you must individually connect to each directory.
- In the curl command, you must use a base64-encoded password in the "**LDAP_BINDPASSWORD**" parameter. To encode the password, use the following command:

```
echo -n <password> | base64
```

Following is an example output:

```
UGFzc3cwcmQ=
```

The sample curl command for creating the LDAP connection resembles the following code:

```
curl -k -X POST 'https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/onboardDirectory'
--header "Authorization: bearer $ACCESS_TOKEN" \
--header 'Content-Type: application/json'
-d
'{
   "LDAP_ID": "msad",
   "LDAP_URL": "ldap://corp.abc.com:389",
   "LDAP_BASEDN": "DC=ibmtest,DC=com",
   "LDAP_BINDDN": "CN=Administrator,cn=Users,DC=ibmtest,DC=com",
   "LDAP_BINDPASSWORD": "password in base64 encoded",
   "LDAP_TYPE": "Microsoft Active Directory",
   "LDAP_USERFILTER": "(&(sAMAccountName=%v)(objectclass=person))",
   "LDAP_GROUPFILTER": "(&(cn=%v)(objectcategory=group))",
   "LDAP_USERIDMAP": "user:sAMAccountName",
   "LDAP_GROUPIDMAP":"*:cn",
   "LDAP_GROUPMEMBERIDMAP": "memberOf:member"
   "LDAP_GROUP_SEARCHBASE_LIST":
["OU=searchbase1,DC=ibmtest,DC=com","OU=searchbase2,DC=ibmtest,DC=com","OU=searchbase3,DC=ibmtest,DC=com"]
   "LDAP_PAGINGSEARCH": "true",
   "LDAP_PAGING_SIZE":"1000"
}'
```

For more information about the LDAP parameters, see Configuring LDAP authentication.

The response resembles the following code:

```
"8b019a10-daa0-11e7-8dba-bf3c83e12db5"
```

Get information about an LDAP directory

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/{ID}

Command

GET

Command output format

application/json

Note:

- There is a limitation on liberty, that is, the **LDAP_USERFILTER** and **LDAP_GROUPFILTER** fields must contain an attribute with **=%v** value. For example, **(cn=%v)** for Cloud Pak and Cloud Pak login flow to work.

- To get information about an LDAP directory, you must assign the directory as a resource to a team. Only then the team members can use this API to get information about the LDAP directory. For more information about how to assign a resource to a team, see Assign resources to a team. The format of the directory resource CRN is **crn:v1:icp:private:iam::::Directory:<ID>**, where **<ID>** is the GUID identifier that is assigned to the connection in the **id** field.

The sample curl command resembles the following code:

```
curl -k -X GET \
  -H "Authorization: Bearer $ACCESS_TOKEN" \
  -H 'Content-Type: application/json' \
  -H 'Accept: application/json' \
  "https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/{ID}"
```

The response resembles the following code:

```
{"id":"69452b20-bb3d-11e8-98b2-
970b1dcdf410","LDAP_ID":"openldap","LDAP_REALM":"REALM","LDAP_HOST":"corp.abc.com","LDAP_PORT":"389","LDAP_IG
NORECASE":"false","LDAP_BASEDN":"dc=ibm,dc=com","LDAP_BINDDN":"cn=admin,dc=ibm,dc=com","LDAP_TYPE":"Custom","
LDAP_USERFILTER":"(&(uid=%v)(objectclass=person))","LDAP_GROUPFILTER":"(&(cn=%v)
(objectclass=groupOfUniqueNames))","LDAP_USERIDMAP":"*:uid","LDAP_GROUPIDMAP":"*:cn","LDAP_GROUPMEMBERIDMAP":
"groupOfUniqueNames:uniquemember","LDAP_URL":"ldap://corp.abc.com:389","LDAP_PROTOCOL":"ldap"}
```

The CRN for this LDAP directory resource is **crn:v1:icp:private:iam::::Directory:69452b20-bb3d-11e8-98b2-970b1dcdf410**.

Update an LDAP directory

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/{id}

Command

PUT

Command output format

application/json

Note: Use **id** to update the LDAP directory. To get the id, see Get information about an LDAP directory.

The sample curl command resembles the following code:

```
curl -k -X PUT \
    -H "Content-type: application/json" \
    -H "Authorization: Bearer ${ACCESS_TOKEN}" \
    "https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/{id}" \
    -d '{
      "LDAP_ID": "openldap",
      "LDAP_URL": "ldap://corp.abc.com:389",
      "LDAP_BASEDN": "dc=ibm,dc=com",
      "LDAP_BINDDN": "cn=admin,dc=ibm,dc=com",
      "LDAP_BINDPASSWORD": "UGFzc3cwcmQ=",
      "LDAP_TYPE": "Custom",
      "LDAP_USERFILTER": "(&(uid=%v)(objectclass=person))",
      "LDAP_GROUPFILTER": "(&(cn=%v)(objectclass=groupOfUniqueNames))",
      "LDAP_USERIDMAP": "*:uid",
      "LDAP_GROUPIDMAP": "*:cn",
      "LDAP_GROUPMEMBERIDMAP": "groupOfUniqueNames:uniquemember"
      "LDAP_GROUP_SEARCHBASE_LIST":
["OU=searchbase1,DC=ibmtest,DC=com","OU=searchbase2,DC=ibmtest,DC=com","OU=searchbase3,DC=ibmtest,DC=com"]
      "LDAP_PAGINGSEARCH": "true",
      "LDAP_PAGING_SIZE":"1000"
    }'
```

The response resembles the following code:

```
{"id":"e02d78b0-72df-11e8-8d5e-
93a06ac1d3fc","LDAP_ID":"openldap","LDAP_REALM":"REALM","LDAP_HOST":"9.37.204.115","LDAP_PORT":"389","LDAP_IG
NORECASE":"false","LDAP_BASEDN":"dc=ibm,dc=com","LDAP_BINDDN":"cn=admin,dc=ibm,dc=com","LDAP_TYPE":"Custom","
LDAP_USERFILTER":"(&(uid=%v)(objectclass=person))","LDAP_GROUPFILTER":"(&(cn=%v)
(objectclass=groupOfUniqueNames))","LDAP_USERIDMAP":"*:uid","LDAP_GROUPIDMAP":"*:cn","LDAP_GROUPMEMBERIDMAP":
"groupOfUniqueNames:uniquemember","LDAP_URL":"ldap://corp.abc.com:389","LDAP_PROTOCOL":"ldap"}
```

List LDAP connections

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/list

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/list'
```

The response resembles the following code:

```
[{"id":"8b019a10-daa0-11e7-8dba-
bf3c83e12db5","LDAP_ID":"Corp","LDAP_REALM":"REALM","LDAP_HOST":"corp.abc.com","LDAP_PORT":"389","LDAP_BASEDN
":"o=ibm.com","LDAP_BINDDN":"","LDAP_BINDPASSWORD":"","LDAP_TYPE":"IBM Tivoli Directory
Server","LDAP_USERFILTER":"(&(emailAddress=%v)(objectclass=ePerson))","LDAP_GROUPFILTER":"(&(cn=%v)
(objectclass=groupOfUniqueNames))","LDAP_USERIDMAP":"*:emailAddress","LDAP_GROUPIDMAP":"*:cn","LDAP_GROUPMEMB
ERIDMAP":"groupOfUniqueNames:uniqueMember","LDAP_URL":"ldap://corp.abc.com:389","LDAP_PROTOCOL":"ldap"}]
```

Delete an LDAP directory

Note: Use this API to delete an LDAP directory when only one LDAP directory is configured.

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/offboardDirectory

Command

POST

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X POST --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/offboardDirectory'
```

The response resembles the following code:

```
"Count: 1"
```

Delete LDAP directory by ID

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/offboardDirectory?id={id}

Command

POST

Command output format

application/json

Note: Use `id` to update the LDAP directory. To get the id, see Get information about an LDAP directory.

The sample curl command resembles the following code:

```
curl -k -X POST \
  -H "Authorization: Bearer $ACCESS_TOKEN" \
  -H 'Content-Type: application/json' \
  "https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/offboardDirectory?id={id}"
```

The response resembles the following code:

```
{"count":1}
```

Search for user groups in your LDAP directory

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/{id}/fetchUserGroups

Command

GET

Command output format

application/json

Note: Use **id** to update the LDAP directory. To get the id, see Get information about an LDAP directory.

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
"https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/{id}/fetchUserGroups?searchString=*sec*"
```

The response resembles the following code:

```
[{"cn":"security","dn":"cn=security,cn=platform,ou=cloud,ou=isl,ou=groups,dc=ibm,dc=com"},
{"cn":"cloudSecurity","dn":"cn=cloudSecurity,ou=cloud,ou=isl,ou=groups,dc=ibm,dc=com"}]
```

Search for users in your LDAP directory

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/{id}/fetchUsers

Command

GET

Command output format

application/json

Note: Use **id** to update the LDAP directory. To get the id, see Get information about an LDAP directory.

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
"https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/{id}/fetchUsers?searchString=*test*"
```

The response resembles the following code:

```
[{"cn":"TestUser","dn":"uid=BDD150908,c=fr,ou=bluepages,o=ibm.com","userId":"test.user@fr.ibm.com"},
{"cn":"test1","dn":"uid=TY11DN554,c=cz,ou=bluepages,o=ibm.com","userId":"test1.user@ibm.com"}]
```

Import user groups from your LDAP directory

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/{id}/importUserGroups

Command

POST

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X POST --header "Authorization: Bearer $ACCESS_TOKEN" --header 'Content-Type: application/json' --
header 'Accept: application/json' -d '{ "baseDN":
"cn=security,cn=platform,ou=cloud,ou=isl,ou=groups,dc=ibm,dc=com" }'
"https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/fb01b1d0-1fa4-11e8-80d6-
15882dd657a0/importUserGroups"
```

The response resembles the following code:

```
{"name":"security","directoryId":"fb01b1d0-1fa4-11e8-80d6-
15882dd657a0","userGroupDN":"cn=security,cn=platform,ou=cloud,ou=isl,ou=groups,dc=ibm,dc=com"}
```

Import users from your LDAP directory

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/directory/ldap/{id}/importUser

Command

POST

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X POST --header "Authorization: Bearer $ACCESS_TOKEN" --header 'Content-Type: application/json' --
header 'Accept: application/json' -d '{ "baseDN": "uid=testuser,ou=people,dc=ibm,dc=com" }'
"https://<cluster_address>/idmgmt/identity/api/v1/directory/ldap/fb01b1d0-1fa4-11e8-80d6-
15882dd657a0/importUser"'
```

The response resembles the following code:

```
{"userId":"testuser","directoryId":"fb01b1d0-1fa4-11e8-80d6-
15882dd657a0","firstName":"TestUser","lastName":"","email":"testuser@ibm.com","lastLogin":"","userBaseDN":"ui
d=testuser,ou=people,dc=ibm,dc=com","type":"LDAP","_id":"testuser","loopback__model__name":"Users"}
```

User group management APIs

APIs to manage user groups.

Base path: **`https://<cluster_address>/idmgmt/identity/api/v1/usergroup`**

Get all user groups

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/usergroup

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/usergroup'
```

The response resembles the following code:

```
[{"name":"security","directoryId":"fb01b1d0-1fa4-11e8-80d6-
15882dd657a0","userGroupDN":"cn=security,cn=platform,ou=cloud,ou=isl,ou=groups,dc=ibm,dc=com"}]
```

Delete a user group

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/usergroup/{id}

Command

DELETE

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X DELETE --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/usergroup/<usergroup name>'
```

The response resembles the following code:

```
{"count":1}
```

User management APIs

APIs to manage users.

Base path: **https://<cluster_address>/idmgmt/identity/api/v1/users**

Get information about all users

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/users

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/users'
```

The output resembles the following code:

```
[{"userId":"aaa","directoryId":"fb01b1d0-1fa4-11e8-80d6-
15882dd657a0","firstName":"AAA","lastName":"","email":"aaa@ibm.com","lastLogin":"","userBaseDN":"uid=aaa,ou=p
eople,dc=ibm,dc=com","type":"LDAP"},{"userId":"testuser","directoryId":"fb01b1d0-1fa4-11e8-80d6-
15882dd657a0","firstName":"TestUser","lastName":"","email":"testuser@ibm.com","lastLogin":"","userBaseDN":"ui
d=testuser,ou=people,dc=ibm,dc=com","type":"LDAP"},{"userId":"bbb","directoryId":"fb01b1d0-1fa4-11e8-80d6-
15882dd657a0","firstName":"BBB","lastName":"","email":"bbb@ibm.com","lastLogin":"","userBaseDN":"uid=bbb,ou=p
eople,dc=ibm,dc=com","type":"LDAP"}]
```

Get information about all members of the team to which a user belongs

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/users?filter={\"{id}\":\"{name}\"}

Command

GET

Command output format

application/json

In the following example, you can get all members of the team to which Tom belongs.

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/users?filter=\{\"id\":\"tom\"\}'
```

The output resembles the following code:

```
[{"userId":"icpuser20","firstName":"Icpuser20","lastName":"","email":"icpuser20@ibm.com","directoryId":"d4b58
be0-3426-11e9-b2dc-0964dfea827a","userBaseDN":"uid=icpuser20,ou=people,dc=ibm,dc=com","roles":
[{"id":"crn:v1:icp:private:iam::::role:Viewer"}]},
{"userId":"tom","firstName":"Tom","lastName":"Sen","email":"tom@ibm.com","directoryId": "e02d78b0-72df-11e8-
8d5e-
93a06ac1d3fc","lastLogin":"","deleted":false,"userBaseDN":"uid=tom,ou=people,dc=ibm,dc=com","type":"LDAP",}]
```

Get the teams to which a user is assigned

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/teams

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl --header 'Content-Type: application/json' --header 'Accept: application/json' -H "Authorization: Bearer
$ACCESS_TOKEN" 'https://<cluster_address>/idmgmt/identity/api/v1/teams' --insecure
```

The output resembles the following code:

```
[{"teamId":"testteam","name":"testteam","users":
[{"userId":"ibmuser1","lastLogin":"","deleted":false,"userBaseDN":"uid=ibmuser1,ou=people,dc=ibm,dc=com","typ
e":"LDAP","roles":[{"id":"crn:v1:icp:private:iam::::role:Administrator"}]},
{"userId":"tom","firstName":"Tom","lastName":"","email":"tom@ibm.com","userBaseDN":"uid=tom,ou=people,dc=ibm,
dc=com","roles":[{"id":"crn:v1:icp:private:iam::::role:Administrator"}]}],"usergroups":[],"resources":
[],"serviceids":[],"directoryList":["9de4fb10-2868-11e9-97f4-8f832f9bc6f9"]}]
```

Get the number of teams to which a user is assigned

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/teams/count

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -X GET --header 'Content-Type: application/json' --header 'Accept: application/json'
--header 'Authorization: Bearer $ACCESS_TOKEN' 'https://<cluster_address>/idmgmt/identity/api/v1/teams/count'
--insecure
```

The output resembles the following code:

```
1
```

Get the directories to which a user has access

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/teams/directories

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -X GET --header 'Content-Type: application/json' --header 'Accept: application/json' --header
"Authorization: Bearer $ACCESS_TOKEN" 'https://<cluster_address>/idmgmt/identity/api/v1/teams/directories' --
insecure
```

The output resembles the following code:

```
[{"id":"311d0730-65b6-11e9-bfc1-
979fb092ad2e","LDAP_ID":"customLdap","LDAP_REALM":"REALM","LDAP_HOST":"9.37.204.115","LDAP_PORT":"389","LDAP_
IGNORECASE":"false","LDAP_BASEDN":"dc=ibm,dc=com","LDAP_BINDDN":"cn=admin,dc=ibm,dc=com","LDAP_TYPE":"Custom"
,"LDAP_USERFILTER":"(&(uid=%v)(objectclass=person))","LDAP_GROUPFILTER":"(&(cn=%v)
(objectclass=groupOfUniqueNames))","LDAP_USERIDMAP":"*:uid","LDAP_GROUPIDMAP":"*:cn","LDAP_GROUPMEMBERIDMAP":
"groupOfUniqueNames:uniqueMember","LDAP_URL":"ldap://9.37.204.115:389","LDAP_PROTOCOL":"ldap"}]
```

Get the highest role that is assigned to a user across teams

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/teams/highestRole

Command

GET

Command output format

application/json

Note: Only the logged in user can view the highest role that is assigned to the user.

The sample curl command resembles the following code:

```
curl -X GET --header 'Content-Type: application/json' --header 'Accept: application/json' --header
'Authorization: Bearer $ACCESS_TOKEN' 'https://<cluster_address>/idmgmt/identity/api/v1/teams/highestRole' --
insecure
```

The output resembles the following code:

```
"Operator"
```

Get the highest role that is assigned to a user and CRN across teams

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idmgmt/identity/api/v1/teams/highestRole?crn={CRN}

Command

GET

Command output format

application/json

Note: Only the logged in user can view the highest role that is assigned to the user.

The sample curl command resembles the following code:

```
curl -X GET --header 'Content-Type: application/json' --header 'Accept: application/json' --header
'Authorization: Bearer $ACCESS_TOKEN' 'https://<cluster_address>/idmgmt/identity/api/v1/teams/highestRole?
```

```
crn=crn%3Av1%3Aicp%3Aprivate%3Ak8%3Amycluster%3An%2Fdefault%3A%3A%3A' --insecure
```

The output resembles the following code:

```
"Operator"
```

Get the team resources that are assigned to a user

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/teams/resources

Command

GET

Command output format

application/json

Note: Only the logged in user can view the resources that are assigned to the user.

The sample curl command resembles the following code:

```
curl --header 'Content-Type: application/json' --header 'Accept: application/json' -H "Authorization: Bearer
$ACCESS_TOKEN" 'https://<cluster_address>/idmgmt/identity/api/v1/teams/resources' --insecure
```

The output resembles the following code:

```
[{"crn":"crn:v1:icp:private:k8:mycluster.icp:n/kube-
system:::","serviceName":"k8","region":"mycluster.icp","namespaceId":"kube-
system","scope":"namespace","actions":"CRUD"}]
```

Get the team resources that are assigned to a user by resource type

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/teams/resources?resourceType={filter-type}

Command

GET

Command output format

application/json

Note: Only the logged in user can view the resources that are assigned to the user.

Valid filter type values are **namespace**, **helm-charts**, **repo**, **clusterserviceclass**, **clusterserviceplan**, and **Directory**.

The sample curl command resembles the following code:

```
curl --header 'Content-Type: application/json' --header 'Accept: application/json' -H "Authorization: Bearer
$ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/teams/resources?resourceType=namespace' --insecure
```

The output resembles the following code:

```
[{"crn":"crn:v1:icp:private:k8:mycluster.icp:n/kube-
system:::","serviceName":"k8","region":"mycluster.icp","namespaceId":"kube-
system","scope":"namespace","actions":"CRUD"}]
```

Get the team resources that are assigned to a user by resource type and action type

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/teams/resources?resourceType={filter-type}&actionType={action-type}

Command

GET

Command output format

application/json

Note: Only the logged in user can view the resources that are assigned to the user.

Valid filter type values are **namespace**, **helm-charts**, **repo**, **clusterserviceclass**, **clusterserviceplan**, and **Directory**.
Valid action type values are **view** and **deploy**.

The sample curl command resembles the following code:

```
curl --header 'Content-Type: application/json' --header 'Accept: application/json' -H "Authorization: Bearer
$ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/teams/resources?resourceType=repo&actionType=deploy' --
insecure
```

The output resembles the following code:

```
[{"crn":"crn:v1:icp:private:helm-catalog:mycluster:r/local-charts::helm-repos:","serviceName":"helm-
catalog","region":"mycluster","resourceType":"helm-repos","repoId":"local-
charts","scope":"repo","actions":"CRUD"}]
```

Get team role mappings

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/teams/roleMappings

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -X GET --header 'Content-Type: application/json' --header 'Accept: application/json' --header
"Authorization: Bearer $ACCESS_TOKEN" 'https://<cluster_address>/idmgmt/identity/api/v1/teams/roleMappings' -
-insecure
```

The output resembles the following code:

```
["icp:testteam:operator","icp:default:member"]
```

Delete a user

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/users/{id}

Command

DELETE

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X DELETE --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/users/aaa'
```

The output resembles the following code:

```
"Count: 1"
```

Get a user's account information

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/users/{user}/getAccounts

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
'https://<cluster_address>/idmgmt/identity/api/v1/users/testuser/getAccounts'
```

The output resembles the following code:

```
{"userId":"testuser","directoryId":"fb01b1d0-1fa4-11e8-80d6-
15882dd657a0","firstName":"TestUser","lastName":"","email":"testuser@ibm.com","lastLogin":"","userBaseDN":"uid=testuser,ou=people,dc=ibm,dc=com","type":"LDAP","url":"/identity/api/v1/users/testuser"}
```

Single sign-on APIs

Use these APIs to configure single sign-on (SSO) in IBM Cloud Pak for AIOps cluster.

The IAM roles, **CloudPakAdministrator** and **Administrator** are allowed to access the SSO APIs. For more information about IAM roles and actions, see Role-based access control (RBAC) for clusters.

To use these APIs, you must add an authorization header to your request. You need an access token to add to the authorization header. To obtain the access token, see Preparing to run component or management API commands.

Enable SAML

The following example shows how to enable SAML using the API:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/v1/saml/management

Command

PUT

Command output format

application/json

The sample curl command resembles the following code:

```
curl -v -k -X PUT --header "Authorization: Bearer $ACCESS_TOKEN" \
--header 'Content-Type: application/json' -d '{"enable": true}'
```

```
https://<cluster_address>/idmgmt/v1/saml/management
```

The response resembles the following code:

```
*   Trying 1.1.1.1...
* Connected to 1.1.1.1 (1.1.1.1) port 8443 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 594 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384
.
.
* ALPN, server accepted to use http/1.1
> PUT /idmgmt/v1/saml/management HTTP/1.1
> Host: 1.1.1.1:8443
> User-Agent: curl/7.47.0
> Accept: */*
> Authorization: Bearer
.
.
> Content-Type: application/json
> Content-Length: 16
>
* upload completely sent off: 16 out of 16 bytes
.
.
* Connection #0 to host 1.1.1.1 left intact

Response:
---------
Configuration successful

root@rise1:~#
```

Export metadata

From foundational services version 3.19.0 and later, you can export the SAML metadata by using the following methods:

1. **samlmetadata** API
2. **cloudctl** command

Note:

- In foundational services version 3.17.0 and earlier, the SAML export metadata is supported by the **cloudctl** command only.
- To export the SAML metadata in foundational services version 3.19.0, you need to login to the cluster and enable SAML by using **cloudctl** command or APIs .
- From foundational services version 3.20, you do not require to enable the SAML to export SAML metadata. You can export the SAML metadata by directly using the **samlmetadata** API.

Details of exporting SAML metadata by using **samlmetadata** API and **cloudctl** command:

Exporting the SAML metadata using `samlmetadata` API

The following example shows how to export SAML metadata by using the **samlmetadata** API:

API version
1.0.0
API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idprovider/v3/saml/metadata/

Command

```
GET
```

Command output format

```
application/xml
```

The sample curl command resembles the following code:

```
curl -k -v -X GET 'https://<cluster_address>/idprovider/v3/saml/metadata/<SERVICE_PROVIDER_ID>' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

For example:

```
curl -k -v -X GET 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v3/saml/metadata/defaultSP' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

Note: As of now, the **defaultSP** is the only supported service provider metadata for the **samlmetadata** API .

The response resembles the following code:

Note: If you choose to download SAML metadata API, the response code shows **200 OK** status with SAML metadata sample.

```
200 OK

<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="https://cp-
console.apps.mycluster.mydomain.com/ibm/saml20/defaultSP">
    <md:SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="true"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <md:KeyDescriptor use="signing">
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>

<ds:X509Certificate>MIIDWjCCAkKgAwIBAgIRALy+U3ooWL4WotnPuKUixTswDQYJKoZIhvcNAQELBQAwHDEaMBgGA1UE
AxMRY3MtY2EtY2VydGlmaWNhdGUwHhcNMjIwNDE4MTIyODI4WhcNMjMwNDE4MTIyODI4WjAdMRsw
GQYDVQQDExJtYW5hZ2VtZW50LWluZ3Jlc3MwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQC6h0rYALFRA104/DGWLlfiuftuoxT+Ab2FzsnmkO1q8Iu/SUhLf5FFnb05BrcemrL+0OJcZEyR
ALdxliWSWPG/I/9Uj+BcVlVdQupND/RQTQHS5ECYEoLJJvHkFHVj7+/WxyN3eYmqlz9OZHKp17f1
t79HDOYPZSKVzhLhZ9MsLM/G3xIF8feRpJRoQSYUUKHB6gDXZ4Kbfui7kd/LMrQK820psA8z/Bjy
dYQ2SXXUo4BfgpUxjN+VM1dGUy0khkuUSdNKlW9pLmZdT1FH7FjWH9zBWOOHz8szr+Pr4ZavHmFL
SUP4nDMpFeH/v6+cBm48LrezWAAUzZuSTYxRQeuhAgMBAAGjgZUwgZIwDgYDVR0PAQH/BAQDAgWg
MBMGA1UdJQQMMAoGCCsGAQUFBwMBMAwGA1UdEwEB/wQCMAAwHwYDVR0jBBgwFoAUzW7sLnme6JsG
DZM/SC2ZRUwS/OswPAYDVR0RBDUwM4IxY3AtY29uc29sZS5hcHBzLnRhbWlsLWJlZHJvY2stZGV2
LmNwLmZ5cmUuaWJtLmNvbTANBgkqhkiG9w0BAQsFAAOCAQEASHgDST++rAi79e0P5TXrUF2QQ7dc
o2pgiv/xNKvwm/URzaSGZ1CYq69hXOd+SFZ0sverFFEQZhBXxKVIDGFMT2Jll189sTtQsQnodbpj
gwQxB7c4KbqI4EKU33jj9lT5CmmzHWtO5cPa7br4nOcPKp7bl2q79/aDbP7GYLMp8sGMacopq5J7
vAkL3O3De3sv2p7wg5nHB0JBA/k7Ecd/34gKpftfwZxNsBAxL+dfwOdHjK1bSlHoyo5F7gjwvLVx
dCGdw0AfSNuw0efyGEK2+bA9bc+pzGQf6vKaaSQpdL/YC6b5QLpQkpculwl+lLESFJkdEzmM2VRB
XfgckB39kg==</ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:KeyDescriptor use="encryption">
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:X509Data>

<ds:X509Certificate>MIIDWjCCAkKgAwIBAgIRALy+U3ooWL4WotnPuKUixTswDQYJKoZIhvcNAQELBQAwHDEaMBgGA1UE
AxMRY3MtY2EtY2VydGlmaWNhdGUwHhcNMjIwNDE4MTIyODI4WhcNMjMwNDE4MTIyODI4WjAdMRsw
GQYDVQQDExJtYW5hZ2VtZW50LWluZ3Jlc3MwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQC6h0rYALFRA104/DGWLlfiuftuoxT+Ab2FzsnmkO1q8Iu/SUhLf5FFnb05BrcemrL+0OJcZEyR
ALdxliWSWPG/I/9Uj+BcVlVdQupND/RQTQHS5ECYEoLJJvHkFHVj7+/WxyN3eYmqlz9OZHKp17f1
t79HDOYPZSKVzhLhZ9MsLM/G3xIF8feRpJRoQSYUUKHB6gDXZ4Kbfui7kd/LMrQK820psA8z/Bjy
dYQ2SXXUo4BfgpUxjN+VM1dGUy0khkuUSdNKlW9pLmZdT1FH7FjWH9zBWOOHz8szr+Pr4ZavHmFL
SUP4nDMpFeH/v6+cBm48LrezWAAUzZuSTYxRQeuhAgMBAAGjgZUwgZIwDgYDVR0PAQH/BAQDAgWg
MBMGA1UdJQQMMAoGCCsGAQUFBwMBMAwGA1UdEwEB/wQCMAAwHwYDVR0jBBgwFoAUzW7sLnme6JsG
DZM/SC2ZRUwS/OswPAYDVR0RBDUwM4IxY3AtY29uc29sZS5hcHBzLnRhbWlsLWJlZHJvY2stZGV2
LmNwLmZ5cmUuaWJtLmNvbTANBgkqhkiG9w0BAQsFAAOCAQEASHgDST++rAi79e0P5TXrUF2QQ7dc
o2pgiv/xNKvwm/URzaSGZ1CYq69hXOd+SFZ0sverFFEQZhBXxKVIDGFMT2Jll189sTtQsQnodbpj
gwQxB7c4KbqI4EKU33jj9lT5CmmzHWtO5cPa7br4nOcPKp7bl2q79/aDbP7GYLMp8sGMacopq5J7
vAkL3O3De3sv2p7wg5nHB0JBA/k7Ecd/34gKpftfwZxNsBAxL+dfwOdHjK1bSlHoyo5F7gjwvLVx
dCGdw0AfSNuw0efyGEK2+bA9bc+pzGQf6vKaaSQpdL/YC6b5QLpQkpculwl+lLESFJkdEzmM2VRB
XfgckB39kg==</ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://cp-console.apps.mycluster.mydomain.com/ibm/saml20/defaultSP/slo"/>
        <md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://cp-console.apps.mycluster.mydomain.com/ibm/saml20/defaultSP/acs" index="0"
```

```
isDefault="true"/>
    </md:SPSSODescriptor>
</md:EntityDescriptor>
```

Error response code

The following table shows the list of error response code:

Table. Error response codes

Error code	Message	Description
404	`The saml id provided by user is not exist in liberty saml db . Please provide valid saml id to get metadata`	If SAML ID is invalid, it throws 404 error with the displayed message.
400	`Insufficient user permission for role:Viewer`	If you are not authorized to download the SAML metadata API, it displays 400 error with the displayed message.
500		Any system error while retrieving SAML status or SAML metadata download, displays 500 error.

Using the `cloudctl` command

When you run the command, a metadata file is downloaded from IBM Cloud Pak for Multicloud Management. The file is saved with the name that you specify. You can upload the file to your enterprise SAML server.

Run the following command to upload the file to your SAML server:

```
cloudctl iam saml-export-metadata --file <file_name>.xml
```

A sample metadata file resembles the following code:

```
<?xml version="1.0" encoding="UTF-8"?><md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" \
entityID="https://mycluster.mydomain.com:8443/ibm/saml20/defaultSP"><md:SPSSODescriptor
AuthnRequestsSigned="true" \
WantAssertionsSigned="true" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"> \
<md:KeyDescriptor use="signing"><ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"><ds:X509Data> \
<ds:X509Certificate>MIID9zCCAd8CCQDIJbZgmPut9DANBgkqhkiG9w0BAQsFADBjMQswCQYDVQQGEwJVUzERMA8GA1UE
.
.
btEmEMpzbGQy8Lb190tLeLZNW2zrBWbRmxzShn9ekS58aEbeD6PBTzWsKXsgYhZWWXw=</ds:X509Certificate> \
</ds:X509Data></ds:KeyInfo></md:KeyDescriptor><md:KeyDescriptor use="encryption"> \
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"><ds:X509Data> \
<ds:X509Certificate>MIID9zCCAd8CCQDIJbZgmPut9DANBgkqhkiG9w0BAQsFADBjMQswCQYDVQQGEwJVUzERMA8GA1UE
.
.
btEmEMpzbGQy8Lb190tLeLZNW2zrBWbRmxzShn9ekS58aEbeD6PBTzWsKXsgYhZWWXw=</ds:X509Certificate></ds:X509Data> \
</ds:KeyInfo></md:KeyDescriptor><md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST" \
Location="https://mycluster.mydomain.com:8443/ibm/saml20/defaultSP/slo"/><md:AssertionConsumerService \
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" \
Location="https://mycluster.mydomain.com:8443/ibm/saml20/defaultSP/acs" index="0" isDefault="true"/>\
</md:SPSSODescriptor></md:EntityDescriptor>
```

Import metadata

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/v1/saml/upload

Command

POST

Command output format

application/json

Note: You use this API call to upload the metadata file that you received from your enterprise SAML server. File name in the example is **samlidp2_IBM_metadata_CIS_STAGE.xml**.

The sample curl command resembles the following code:

```
curl -v -k -X POST --header "Authorization: Bearer $ACCESS_TOKEN" \
-F 'data=@samlidp2_IBM_metadata_CIS_STAGE.xml' https://<cluster_address>/idmgmt/v1/saml/upload
```

The sample response resembles the following code:

```
*   Trying 1.1.1.1...
* TCP_NODELAY set
.
.
* ALPN, offering http/1.1
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
*   CAfile: C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
  CApath: none
* TLSv1.2 (OUT), TLS header, Certificate Status (22):
} [5 bytes data]
.
.
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server accepted to use http/1.1
* Server certificate:
.
.
> POST /idmgmt/v1/saml/upload HTTP/1.1
> Host: 1.1.1.1:8443
> User-Agent: curl/7.53.0
> Accept: */*
> Authorization: Bearer
.
.
> Content-Length: 5313
> Expect: 100-continue
> Content-Type: multipart/form-data; boundary=------------------------9ecb5200d6d7e5e6
.
.
< HTTP/1.1 200 OK
.
.
{ [31 bytes data]
100  5344  100    31  100  5313      1    275  0:00:31  0:00:19  0:00:12     0Metadata uploaded successfully.
```

Verify SSO configuration status

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/v1/saml/status

Command

GET

Command output format

```
application/json
```

Any of the following responses is valid:

```
{"enable":true}
SAML is enabled and the metadata file is successfully uploaded.

{"status":false,"description":`["SAML Feature not enabled"]`}
SAML is not enabled.

{"status":false,"description":`["IDP Metadata not uploaded"]`}
SAML is enabled but the metadata file that you received from your enterprise SAML server is not uploaded.

{"status":false,"description":`["SAML Feature not enabled", "IDP Metadata not uploaded"]`}
SAML is not enabled, or you did not upload the metadata file that you received from your enterprise SAML
server.
```

The sample curl command resembles the following code:

```
curl -k -v -X GET --header "Authorization: Bearer $ACCESS_TOKEN"
https://<cluster_address>/idmgmt/v1/saml/status
```

The sample response resembles the following code:

```
*   Trying 2.2.2.2...
* Connected to 2.2.2.2 (2.2.2.2) port 8443 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 594 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384
.
.
* ALPN, server accepted to use http/1.1
> GET /idmgmt/v1/saml/status HTTP/1.1
> Host: 2.2.2.2:8443
> User-Agent: curl/7.47.0
> Accept: */*
> Authorization: Bearer
.
.
< HTTP/1.1 200 OK
< Server: openresty/1.11.2.4
< Date: Tue, 24 Jul 2018 08:43:00 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 16
< Connection: keep-alive
< Vary: Origin, Accept-Encoding
< Access-Control-Allow-Credentials: true
< X-XSS-Protection: 1; mode=block
< X-Frame-Options: SAMEORIGIN
< Strict-Transport-Security: max-age=15552000; includeSubDomains
< X-Download-Options: noopen
< X-Content-Type-Options: nosniff
< X-DNS-Prefetch-Control: off
< ETag: W/"10-H+tYev0PkZ4BjVuzAcmKAO7d1nc"
< X-Frame-Options: SAMEORIGIN
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
<
* Connection #0 to host 2.2.2.2 left intact

_Response:_

{"status":false,"description":["SAML Feature not enabled"]}
```

Disable SAML

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/v1/saml/management

Command

PUT

Command output format

application/json

This API disables SAML and deletes the metadata file that was sent by your enterprise SAML server.

The sample curl command resembles the following code:

```
curl -v -k -X PUT --header "Authorization: Bearer $ACCESS_TOKEN"
--header 'Content-Type: application/json' \
-d '{"enable": false}'  https://<cluster_address>/idmgmt/v1/saml/management
```

The sample response resembles the following code:

```
*   Trying 1.1.1.1...
* Connected to 1.1.1.1 (1.1.1.1) port 8443 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 594 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_256_GCM_SHA384
.
.
* ALPN, server accepted to use http/1.1
> PUT /idmgmt/v1/saml/management HTTP/1.1
> Host: 1.1.1.1:8443
> User-Agent: curl/7.47.0
> Accept: */*
> Authorization: Bearer
.
.
> Content-Type: application/json
> Content-Length: 17
>
* upload completely sent off: 17 out of 17 bytes
< HTTP/1.1 200 OK
< Server: openresty/1.11.2.4
< Date: Sat, 04 Aug 2018 09:54:14 GMT
< Content-Type: text/html; charset=utf-8
< Content-Length: 24
< Connection: keep-alive
< Vary: Origin, Accept-Encoding
< Access-Control-Allow-Credentials: true
< X-XSS-Protection: 1; mode=block
< X-Frame-Options: SAMEORIGIN
< Strict-Transport-Security: max-age=15552000; includeSubDomains
< X-Download-Options: noopen
< X-Content-Type-Options: nosniff
< X-DNS-Prefetch-Control: off
< ETag: W/"18-Akebq37pOvmw0KQ/7FxMJOPs24g"
< X-Frame-Options: SAMEORIGIN
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
<
* Connection #0 to host 1.1.1.1 left intact
Configuration successfulroot@rise1:~#
```

Identity Provider APIs

Use Identity Provider(IdP) APIs to register IdP configuration and their connections.

It is a mandatory step in the following scenarios:

- If you are using foundational services version 3.12.0 or later.
- If you are configuring SSO (single sign-on).

Note:

- Currently, IBM supports SAML only as supported protocol with the V2 version. In future, IdP V2 version will be deactivated. It is recommended to configure SAML from IBM Cloud Pak for AIOps UI. From foundational services version 3.20, you can configure SAML by using IBM Cloud Pak for AIOps UI. For more information, see Configuring single sign-on using IBM Cloud Pak for AIOps UI.
- You can get the hostname from the route available in **ibm-common-service** namespace or where the foundational services is installed. To get the host details, see Getting the host details.
- The IAM roles, **CloudPakAdministrator** and **Administrator** are allowed to access the IdP APIs. For more information about IAM roles and actions, see Role-based access control (RBAC) for clusters.

Register IdP

You can register the IdP by the following methods:

- SAML with SCIM dependency registration
- SAML registration without any dependency
- SAML with LDAP dependency registration

SAML with SCIM dependency registration

The following example shows the IdP registration with ISV (IBM Security Verify) which supports SCIM enabled IdP with SAML. Hence, the **token_attribute_mappings** and **scim_attribute_mappings** can be different for the respective IdPs.

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v2/auth/idsource/registration/

Command

POST

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V2.

The sample curl command resembles the following code:

```
curl -k -X POST 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
--data-raw
'{
    "name": "idp_saml_isv",
    "description": "This is a saml isv config",
    "protocol": "saml",
    "idp_type": "isv",
    "scim": "yes",
    "scim_base_path": "https://bedrock-iam.verify.ibm.com/v2.0/",
    "token_attribute_mappings": {"uid":"userID","first_name":"given_name","last_name":"family_name",
"groups": "groupIds", "email":"email"},
    "jit": "no",
    "scim_attribute_mappings":{
         "user":{
         "principalName":"userName",
         "givenName":"name.givenName",
         "middleName":"name.middleName",
         "familyName":"name.familyName",
         "formatted":"name.formatted"
```

```
        },
        "group":{
         "principalName":"displayName",
         "created":"meta.created",
         "lastModified":"meta.lastModified"
        }
    },
    "config": {
        "grant_type": "client_credentials",
        "token_url": "https://bedrock-iam.verify.ibm.com/v1.0/endpoint/default/token",
        "client_id": "c2eae9d2-900d-4102-840b-023af12e36c5",
        "client_secret":"Your_isv_client_id"
    },
    "status": "enabled"
}'
```

The response resembles the following code:

```
{"status":"Client Successfully registered."}
```

The error message resembles the following code:

```
400 "error": "Insufficient user permission for role : rolename"
400 "error": "schema error:schema validation error followed by error"
400 "error": "duplicate : Idp with protocol=saml is already created"
500 "error": "Other errors"
201 OK Response
```

SAML registration without any dependency

The following example shows the IdP registration by using SAML without any dependency:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v2/auth/idsource/registration/

Command

POST

Command output format

application/json

Note:

- User can find the SAML attribute and claim names from the IdP administration. Specify the SAML attributes and claim names as the values in the **`token_attribute_mappings`** attributes.
- The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V2.

The following sample curl command shows the SAML registration with w3id SAML provider:

```
curl -k -X POST 'https://cp-
console.apps.mycluster.mydomain.myibm.com/idprovider/v2/auth/idsource/registration' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
--data-raw
'{
    "name": "w3id-sample-saml",
    "description": "w3id-sample-saml-test",
    "protocol": "saml",
    "idp_type": "default",
```

```
    "scim": "no",
    "saml_ldap": "None",
    "jit": "yes",
    "token_attribute_mappings": {"uid": "uid","first_name": "firstName","last_name": "lastName", "groups":
"blueGroups", "email": "emailAddress"},
    "status": "enabled"
}'
```

The response resembles the following code:

```
{"status":"Client Successfully registered."}
```

The sample registration for configuring Azure AD with auto-onboarding:

```
curl -k -X POST 'https://cp-console.apps.mycluster.mydomain.com /idprovider/v2/auth/idsource/registration' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
--data-raw
'{
    "name": "idp_saml_azure",
    "description": "This is a saml azure config",
    "protocol": "saml",
    "idp_type": "default",
    "jit": "yes",
    "saml_ldap": "None",
    "token_attribute_mappings": {
            "uid": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name",
            "first_name": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname",
            "last_name": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname",
            "groups": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress",
            "email": "http://schemas.microsoft.com/ws/2008/06/identity/claims/groups"
    },
    "status": "enabled"
}'
```

The error message resembles the following code:

```
400 "error": "Insufficient user permission for role : rolename"
400 "error": "schema error:schema validation error followed by error"
400 "error": "duplicate : Idp with protocol=saml is already created"
500 "error": "Other errors"
201 OK Response
```

SAML with LDAP dependency registration

The following example is provided for IBM W3 SAML Provider and with dependency of an IBM Tivoli Directory Server LDAP. The **token_attribute_mappings** might change based on the SAML provider.

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v2/auth/idsource/registration/

Command

POST

Command output format

application/json

Note:

- User can find the SAML attributes and claim names from the IdP administration. Specify the SAML attributes and claim names as the values in the **token_attribute_mappings** attributes.
- The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V2.

The sample curl command resembles the following code:

```
curl -k -X POST 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
--data-raw
'{
    "name": "saml-test",
    "description": "saml bluepages with ldap",
    "protocol": "saml",
    "idp_type": "default",
    "saml_ldap":"IBM Tivoli Directory Server",
    "token_attribute_mappings": {"uid":"uid", "first_name":"firstName", "last_name":"lastName", "groups":
"blueGroups","email":"email"},
    "jit": "no",
    "status": "enabled"
}'
```

The response resembles the following code:

```
{"status":"Client Successfully registered."}
```

The error message resembles the following code:

```
400 "error": "Insufficient user permission for role : rolename"
400 "error": "schema error:schema validation error followed by error"
400 "error": "duplicate : Idp with protocol=saml is already created"
500 "error": "Other errors"
201 OK Response
```

Get IdP registration

You can get the IdP registration by the following ways:

- By name
- By list
- By query

Get IdP registration by name

The following example shows how to get the details of IdP registration by name:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v2/auth/idsource/registration/idpname

Command

GET

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V2.

The sample curl command resembles the following code:

```
curl -k -X GET 'https://cp-
console.apps.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration/isvbedrock-test' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{
    "name": "isvbedrock-test",
    "description": "ISV bedrock test ISV",
    "protocol": "saml",
    "idp_type": "isv",
    "scim": "yes",
    "scim_base_path": "https://bedrock-iam.verify.ibm.com/v2.0/",
    "token_attribute_mappings": {
        "uid": "userID",
        "first_name": "given_name",
        "last_name": "family_name",
        "groups": "groupIds",
        "email": "email"
    },
    "jit": "no",
    "scim_attribute_mappings": {
        "user": {
            "principalName": "userName",
            "givenName": "name.givenName",
            "middleName": "name.middleName",
            "familyName": "name.familyName",
            "formatted": "name.formatted"
        },
        "group": {
            "principalName": "displayName",
            "created": "meta.created",
            "lastModified": "meta.lastModified"
        }
    },
    "config": {
        "grant_type": "client_credentials",
        "token_url": "https://bedrock-iam.verify.ibm.com/v1.0/endpoint/default/token",
        "client_id": "c2eae9d2-900d-4102-840b-023af12e36c5"
    },
    "status": "enabled"
}
```

The error message resembles the following code:

```
403 "error": "Insufficient user permission for role : rolename"
404 "error": "Document not found"
500 "error": "Other errors"
200 OK Response
```

Get the list of IdP registration

The following example shows how to get the list of IdP registration:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v2/auth/idsource/registration

Command

GET

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V2.

The sample curl command resembles the following code:

```
curl -k -X GET 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration' \
---header "Authorization: Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{
"idp": [
        {
            "name": "isvbedrock-test",
            "description": "ISV bedrock test ISV",
            "protocol": "saml",
            "idp_type": "isv",
            "scim": "yes",
            "scim_base_path": "https://bedrock-iam.verify.ibm.com/v2.0/",
            "token_attribute_mappings": {
                "uid": "userID",
                "first_name": "given_name",
                "last_name": "family_name",
                "groups": "groupIds",
                "email": "email"
            },
            "jit": "no",
            "scim_attribute_mappings": {
                "user": {
                    "principalName": "userName",
                    "givenName": "name.givenName",
                    "middleName": "name.middleName",
                    "familyName": "name.familyName",
                    "formatted": "name.formatted"
                },
                "group": {
                    "principalName": "displayName",
                    "created": "meta.created",
                    "lastModified": "meta.lastModified"
                }
            },
            "config": {
                "grant_type": "client_credentials",
                "token_url": "https://bedrock-iam.verify.ibm.com/v1.0/endpoint/default/token",
                "client_id": "c2eae9d2-900d-4102-840b-023af12e36c5"
            },
            "status": "enabled"
        }
    ]
}
```

The error message resembles the following code:

```
403 "error": "Insufficient user permission for role : rolename"
500 "error": "Other errors"
200 OK Response
```

Get IdP registration by query

The following example shows how to get the IdP registration by query:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

```
idprovider/v2/auth/idsource/registration/
```

Command

```
GET
```

Command output format

```
application/json
```

The sample curl command resembles the following code:

```
curl -k -X GET 'https://cp-console.apps.iam-
test.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration?idp_type=isv' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

You can also alter the query based on the requirements. For example,

```
curl -k -X GET 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration?
name=isvbedrock-test' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

```
curl -k -X GET 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration?
protocol=saml \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

Note: The code contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V2.

The response resembles the following code:

```
    "idp": [
        {
            "name": "isvbedrock-test",
            "description": "ISV bedrock test ISV",
            "protocol": "saml",
            "idp_type": "isv",
            "scim": "yes",
            "scim_base_path": "https://bedrock-iam.verify.ibm.com/v2.0/",
            "token_attribute_mappings": {
                "uid": "userID",
                "first_name": "given_name",
                "last_name": "family_name",
                "groups": "groupIds",
                "email": "email"
            },
            "jit": "no",
            "scim_attribute_mappings": {
                "user": {
                    "principalName": "userName",
                    "givenName": "name.givenName",
                    "middleName": "name.middleName",
                    "familyName": "name.familyName",
                    "formatted": "name.formatted"
                },
                "group": {
                    "principalName": "displayName",
                    "created": "meta.created",
                    "lastModified": "meta.lastModified"
                }
            },
            "config": {
                "grant_type": "client_credentials",
                "token_url": "https://bedrock-iam.verify.ibm.com/v1.0/endpoint/default/token",
                "client_id": "c2eae9d2-900d-4102-840b-023af12e36c5"
            },
            "status": "enabled"
        }
    ]
}
```

The error message resembles the following code:

```
403 "error": "Insufficient user permission for role : rolename"
500 "error": "Other errors"
200 OK Response
```

Updating IdP

To update the IdP registration, you require the IdP name. To get the IdP name, see Get the list of IdP registration. The following example shows how to update the IdP registration:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v2/auth/idsource/registration/

Command

PUT

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V2.

The sample curl command resembles the following code:

```
curl -k -X PUT 'https://cp-
console.apps.sashay.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration/saml-test' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
--data-raw
'{
    "name": "saml-test",
    "description": "saml bluepages no ldap",
    "protocol": "saml",
    "idp_type": "default",
    "saml_ldap":"Custom",
    "token_attribute_mappings": {"uid":"uid", "first_name":"firstName", "last_name":"lastName", "groups":
"blueGroups","email":"email"},
    "jit": "no",
    "status": "enabled"
}  '
```

The response resembles the following code:

```
{"status":"Client Successfully updated."}
```

The error message resembles the following code:

```
400 "error": "Insufficient user permission for role : rolename"
400 "error": "schema constraint error: schema validation error followed by error"
500 "error": "Other errors"
202 OK Response
```

Deleting IdP registration

To delete the IdP registration, you require the IdP name. To get the IdP name, see Get the list of IdP registration. The following example shows how to delete the IdP registration:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v2/auth/idsource/registration/

Command

DELETE

Command output format

application/json

The sample curl command resembles the following code:

```
 curl -k -X DELETE 'https://cp-
console.apps.mycluster.mydomain.com/idprovider/v2/auth/idsource/registration/saml-test'  \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{"status":"Client Successfully deleted.”}
```

The error message resembles the following code:

```
400 "error": "Insufficient user permission for role : rolename"
404 "error": "Document not found"
500 "error": "Other errors"
202 OK Response
```

IdP V3 APIs

- From foundational services version 3.19 and later, you can register OIDC provider by using the IdP V3 API. For more information, see Registering the OIDC clients using IdP V3.
- From foundational services version 3.20 and later, you can register SAML provider by using the IdP V3 API. To register SAML provider using IdP V3, see Registering the SAML clients using IdP V3. It is recommended to configure the SAML from IBM Cloud Pak for AIOps UI. For more information, see Configuring single sign-on using IBM Cloud Pak for AIOps UI.

Registering the OIDC clients using IdP V3

Before you begin

Login to any of the following platforms by using the provided IdP links and register yourself in the application. While registering, use application url as cp-console url and redirect URL as **`https://<cp-console-url>/ibm/api/social-login/redirect/<name of the oidc>`**. Once you are registered, you can get the unique client ID, client secret and **`discovery_url`** endpoint. Currently, only these platforms are verified to register the OIDC by using the IdP V3 API.

Verified platforms to register OIDC provider using IdP V3

Platform	IdP link
IBM Security Verify	https://www.ibm.com/docs/en/erqa?topic=using-security-verify-as-oidc-provider **Note:** Whenever you perform the attributes mapping through the OIDC provider in the ISV application, enable the **Send all known user attributes in the ID token** option. And, ISV groups must not contain spaces.
Google Cloud Platform	https://cloud.google.com/identity-platform/docs/web/google
Microsoft	https://cloud.google.com/identity-platform/docs/web/microsoft
Okta	https://help.okta.com/en-us/Content/Topics/Apps/Apps_App_Integration_Wizard_OIDC.htm

Note: The IdP links for the listed verified platforms can be changed anytime by their service providers. If any of the listed IdP links are broken, check the IdP links on their official websites.

Registering the OIDC clients

The following example shows how to register the client by using IdP V3:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster address

Port number

Cluster Port

Path

idprovider/v3/auth/idsource

Command

POST

Command output format

application/json

Note:

- The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.
- Using the POST call, you can get the unique identifier (UID) as a successful response. The UID is required for the further operations or tasks. Alternatively, to get the list of registered IdP's UID, see Getting the list of registered OIDC clients by query.

The sample curl command to register the client:

```
curl --insecure POST 'https://cp-console.apps.vij-soc.mycluster.mydomain.com/idprovider/v3/auth/idsource/'
--header "Authorization: Bearer $ACCESS_TOKEN"
--header 'Content-Type: application/json'
--data-raw
'{
    "name": "oidc_isv",
    "description": "This is a oidc config for isv",
    "protocol": "oidc",
    "type": "IBMVerify",
    "idp_config": {
        "discovery_url": "https://bedrock-iam.verify.ibm.com/oidc/endpoint/default/.well-known/openid-
configuration",
        "client_id": "4ad41ddc-5f10-4d84-a24f-fb776607895e",
        "client_secret": "rqJMyEgee",
        "token_attribute_mappings": {
            "groups": "groupIds",
            "given_name": "given_name",
            "family_name": "family_name",
            "sub": "uid",
            "email": "email"
        }
    }
}'
```

The response resembles the following code:

```
{
    "status": "success",
    "message": "Identity provider {oidc_isv} is successfully registered with unique identifier
NpiV6qBhsSDEjKRvJDQLl"
}
```

The error message resembles the following code:

```
400 "error":  "E11000 duplicate key error collection: platform-db.cloudpak_ibmid_v3 index: name_1 dup key: {
: <oidc name>" }"
500 "error": "Other errors"
401 "error": "Insufficient user permission for role :Viewer"
202 OK Response
```

Deleting the registration of the client

The following example shows how to delete the registered client by using IdP V3:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster address

Port number

Cluster Port

Path

idprovider/v3/auth/idsource/

Command

DELETE

Command output format

application/json

The sample curl command to delete the client registration:

```
curl --insecure DELETE 'https://cp-console.apps.vij-
soc.mycluster.mydomain.com/idprovider/v3/auth/idsource/NpiV6qBhsSDEjKRvJDQLl' --header "Authorization: Bearer
$ACCESS_TOKEN"
```

The response resembles the following code:

```
{
    "status": "success",
    "message": "{NpiV6qBhsSDEjKRvJDQLl} is deleted"
}
```

The error message resembles the following code:

```
400 "error": "Cannot find {oidc uid}"
401 "error": "Insufficient user permission for role :Viewer"
200: Ok Response
```

Getting the list of registered OIDC clients

You can get the list of registered OIDC clients by using either of the following methods:

- By unique identifier (UID)
- By query

Getting the list of registered OIDC clients by UID

The following example shows how to get the list of all the registered clients by UID:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource/

Command

GET

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.

The following sample curl command to get the list of registered clients by UID:

```
curl --insecure --request GET 'https://cp-
console.mycluster.mydomain.com/idprovider/v3/auth/idsource/yOLi5oHo3t-es4aJFLkC8' --header "Authorization:
Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{
  "name": "oidc_1",
  "description": "This is a oidc config",
  "protocol": "oidc",
  "type": "IBMVerify",
  "idp_config": {
    "discovery_url": "https://bedrock-iam.verify.ibm.com/oidc/endpoint/default/.well-known/openid-
configuration",
    "client_id": "My_Client_ID",
    "client_secret": "My_Client_Secret",
    "token_attribute_mappings": {
      "groups": "groupIds"
    }
  },
  "uid": "NpiV6qBhsSDEjKRvJDQLl"
}
```

The error message resembles the following code:

```
400 "error": "Cannot find {oidc uid}"
401 "error": "Insufficient user permission for role :Viewer"
200: Ok Response
```

Getting the list of registered OIDC clients by query

The following example shows how to get the list of all the registered clients by query:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource/

Command

GET

Command output format

application/json

The following sample curl command to get the list of registered clients by query:

```
curl --insecure --request GET 'https://cp-console.mycluster.mydomain.com/idprovider/v3/auth/idsource?
protocol=oidc' --header "Authorization: Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{
    "idp": [
        {
            "name": "oidc_google",
            "description": "This is a oidc config for google",
```

```
            "protocol": "oidc",
            "type": "google",
            "idp_config": {
                "discovery_url": "https://accounts.google.com/.well-known/openid-configuration",
                "client_id": "My_Client_ID",
                "client_secret": "My_Client_Secret",
                "token_attribute_mappings": {
                    "groups": "groupIds",
                    "given_name": "firstName",
                    "family_name": "lastName",
                    "sub": "uid",
                    "email": "email",
                    "uniqueSecurityName": "uniqueSecurityName"
                }
            },
            "uid": "Uo4Y31mV7unaj5NtyfpIY"
        },
        {
            "name": "oidc_isv",
            "description": "This is a oidc config",
            "protocol": "oidc",
            "type": "IBMVerify",
            "idp_config": {
                "discovery_url": "https://bedrock-iam.verify.ibm.com/oidc/endpoint/default/.well-
known/openid-configuration",
                "client_id": "My_Client_ID",
                "client_secret": "My_Client_Secret",
                "token_attribute_mappings": {
                    "groups": "groupIds",
                    "given_name": "given_name",
                    "family_name": "family_name",
                    "sub": "uid",
                    "email": "email",
                    "uniqueSecurityName": "uniqueSecurityName",
                    "preferred_username": "preferred_username",
                    "displayName": "displayName"
                }
            },
            "uid": "NpiV6qBhsSDEjKRvJDQLl"
        }
    ]
}
```

The error message resembles the following code:

```
400 "error": "Cannot find {oidc uid}"
401 "error": "Insufficient user permission for role :Viewer"
200: Ok Response
```

Updating the OIDC clients

The following example shows how to update the OIDC client by using IdP V3:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster address

Port number

Cluster Port

Path

idprovider/v3/auth/idsource

Command

PUT

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.

The sample curl command to update the client:

```
curl --insecure PUT 'https://cp-
console.mycluster.mydomain.com/idprovider/v3/auth/idsource/NpiV6qBhsSDEjKRvJDQLl'
--header "Authorization: Bearer $ACCESS_TOKEN"
--header 'Content-Type: application/json'
--data-raw
'{
    "name": "oidc_isv",
    "description": "This is a oidc config for isv",
    "protocol": "oidc",
    "type": "IBMVerify",
    "idp_config": {
        "discovery_url": "https://bedrock-iam.verify.ibm.com/oidc/endpoint/default/.well-known/openid-
configuration",
        "client_id": "4ad41ddc-5f10-4d84-a24f-fb776607895e",
        "client_secret": "rqJMyEgee",
        "token_attribute_mappings": {
            "groups": "groupIds",
            "given_name": "given_name",
            "family_name": "family_name",
            "sub": "uid",
            "email": "email"
        }
    }
}'
```

The response resembles the following code:

```
{
    "status": "success",
    "message": "{NpiV6qBhsSDEjKRvJDQLl} is Updated."
}
```

The error message resembles the following code:

```
400 "error": "Document not found"
401 "error": "Insufficient user permission for role :Viewer"
200: Ok Response
```

Registering the SAML clients using IdP V3

You can register the SAML IdP by the following methods:

- SAML with SCIM dependency registration
- SAML registration without any dependency
- SAML with LDAP dependency registration

Note: You can configure only one SAML registration at an instance. The unique identifier generated on the POST request is **`defaultSP`**. For **`idp_metadata`**, you need to provide **`base64`** encoded value of the generated identity provider metadata **`xml`** file.

Get the base64 encoded value of generated identity provider metadata xml file by using the following code:

```
cat idp-metadata-xml |base64
```

SAML with SCIM dependency registration (IdP V3 registration with IBM Security Verify and Okta)

- IdP V3 registration with IBM Security Verify (ISV)
- IdP V3 registration with Okta

IdP V3 registration with IBM Security Verify (ISV)

The following example shows the IdP V3 registration with ISV that supports SCIM enabled IdP with SAML. Hence, the **`token_attribute_mappings`** and **`scim_attribute_mappings`** can be different for the respective IdPs.

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource/registration/

Command

POST

Command output format

application/json

Note:

- The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.
- Using the POST call, you can get the unique identifier (UID) as a successful response. The UID is required for the further operations or tasks. Alternatively, to get the list of registered IdP's UID, see Getting SAML registration by query .

The sample curl command resembles the following code:

```
curl --insecure POST 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v3/auth/idsource'
--header "Authorization: Bearer $ACCESS_TOKEN"
--header 'Content-Type: application/json'
--data-raw
'{
  "name": "idp_saml_isv",
  "description": "This is a saml isv config",
  "protocol": "saml",
  "type": "isv",
  "idp_config": {
    "token_attribute_mappings": {
      "sub": "userID",
      "given_name": "given_name",
      "family_name": "family_name",
      "groups": "groupIds",
      "email": "email"
    },
    "idp_metadata": "<base64-encoded-idp-xml>"
  },
  "scim_config": {
    "scim_base_path": "https://bedrock-iam.verify.ibm.com/v2.0/",
    "grant_type": "client_credentials",
    "token_url": "https://bedrock-iam.verify.ibm.com/v1.0/endpoint/default/token",
    "client_id": "Your_isv_client_id",
    "client_secret": "Your_isv_client_secret",
    "scim_attribute_mappings": {
      "user": {
        "principalName": "userName",
        "givenName": "name.givenName",
        "middleName": "name.middleName",
        "familyName": "name.familyName",
        "formatted": "name.formatted"
      },
      "group": {
        "principalName": "displayName",
        "created": "meta.created",
        "lastModified": "meta.lastModified"
      }
    }
  },
  "jit": "false"
}'
```

The response resembles the following code:

```
{
    "status": "success",
    "message": "Identity provider {idp_saml_isv} is successfully registered with unique identifier defaultSP"
}
```

The error message resembles the following code:

```
401 "error": "Insufficient user permission for role : rolename"
400 "error": "schema error:schema validation error followed by error"
```

```
400 "error": "duplicate : Idp with protocol=saml is already created"
500 "error": "Other errors"
200 OK Response
```

IdP V3 registration with Okta

The following example shows the IdP V3 registration with Okta.

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource/registration/

Command

POST

Command output format

application/json

Note:

- The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.
- Using the POST call, you can get the unique identifier (UID) as a successful response. The UID is required for the further operations or tasks. Alternatively, to get the list of registered IdP's UID, see Getting SAML registration by query .

```
curl --location --request POST 'https://cpconsoleroute/idprovider/v3/auth/idsource' \
--header 'Authorization: Bearer $accessToken' \
--header 'Content-Type: application/json' \
--data-raw '{
            "name": "OKTASAML",
            "description": "OKTASAML",
            "protocol": "saml",
            "type": "okta",
            "idp_config": {
                "token_attribute_mappings": {
                    "sub": "uid",
                    "given_name": "firstName",
                    "family_name": "lastName",
                    "groups": "groups",
                    "email": "emailAddress",
                    "first_name": "firstName",
                    "last_name": "lastName"
                },
                "idp_metadata": "Base64-encoded-metadata"
            },
            "jit": false,
            "scim_config": {
                "redirect_url": ["okta_redirect_url"],
                "scim_attribute_mappings": {
                    "user": {
                        "principalName": "userName",
                        "name": {
                            "givenName": "givenName",
                            "familyName": "familyName"
                        },
                        "displayName": "displayName",
                        "emails": [
                            {
                                "value": "emails",
                                "type": "home"
                            }
```

```
                        ],
                        "id": "id",
                        "userName": "userName"
                    },
                    "group": {
                        "principalName": "displayName",
                        "id": "displayName"
                    }
                }
            }
        }'
```

The response resembles the following:

```
{
    "status": "success",
    "message": "Identity provider {oktasaml} is successfully registered with unique identifier : defaultSP ,
oktaclient created with clientId: yx3a5pwtdrmn0sf8ockyf97dg55lswo7 , client secret: okta-oidcclient-secret"
}
```

The error message resembles the following:

```
401 "error": "Insufficient user permission for role : rolename"
400 "error": "schema error:schema validation error followed by error"
400 "error": "duplicate : Idp with protocol=saml is already created"
500 "error": "Other errors"
200 OK Response
```

token_attribute_mappings: `token_attribute_mappings` is a custom mapping that user can provide based on the SAML attribute mapping that is configured at Okta. User should modify the values, not the key, based on the Okta SAML attribute configuration. For more information about attributes, see Step 9 in Configure CloudPak as SAML service provider (SP) at OKTA.

For more information, see Okta doc for SAML attributes.

scim_attribute_mappings: Provide the mapping attributes as in the CURL command. The defined mapping will be used for filter query, creating attributes and the response to be sent.

redirect_url: `redirect_url` is only applicable for the IdP **"type": "okta"**. The `redirect_url` is used to create the OIDC client registration for the OKTA instance. To get the `redirect_url` while using non IBM OIN App:

```
Based on the environment (a standard org, an EMEA org, and an Okta Preview org), the `redirect_uri` can be
one of the following:
```

- https://system-admin.okta.com/admin/app/cpc/${appName}/oauth/callback
- https://system-admin.okta-emea.com/admin/app/cpc/${appName}/oauth/callback
- https://system-admin.oktapreview.com/admin/app/cpc/${appName}/oauth/callback

  Note: Replace the **${appName}** with the variable name for your application instance. If you don't know the variable name for application instance, you can find it as part of the App Embed link from the browser by opening the embed link or you can also check the **Name** attribute for the application at the **`/Apps API`** endpoint.

  For example, if the URL embedded link is https://dev-68798908-admin.okta.com/admin/app/dev-68798908_iamsaml_3/instance/0oa6vnyhkm86B4xha5d7/#tab-general, the **${appName}** will be **dev-68798908_iamsaml_3** and so the redirect_uri is https://system-admin.okta.com/admin/app/cpc/dev-68798908_iamsaml_3/oauth/callback. For more information, see Okta documentation.

SAML registration without any dependency

The following example shows the IdP V3 registration by using SAML without any dependency:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource/registration/

Command

POST

Command output format

application/json

Note:

- User can find the SAML attribute and claim names from the IdP administration. Specify the SAML attributes and claim names as the values in the **token_attribute_mappings** attributes.
- The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.
- You can also provide an optional attribute called **uniqueSecurityName** in **token_attribute_mappings** payload to map the IdP attributes. By default, **uniqueSecurityName** attribute is mapped to the **Subject** in the response of userinfo/introspect endpoint.

The following sample curl command shows the SAML registration with w3id SAML provider:

```
curl -k -X POST 'https://cp-console.apps.mycluster.mydomain.com/ idprovider/v3/auth/idsource' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
--data-raw
'{
  "name": "w3id-sample-saml",
  "description": "w3id-sample-saml-test",
  "protocol": "saml",
  "type": "default",
  "idp_config": {
    "token_attribute_mappings": {
      "sub": "uid",
      "given_name": "firstName",
      "family_name": "lastName",
      "groups": "blueGroups",
      "email": "emailAddress"
    },
    "idp_metadata": "<base64-encoded-idp-xml>"
  },
  "jit": true
}'
```

The response resembles the following code:

```
{
    "status": "success",
    "message": "Identity provider {w3id-sample-saml} is successfully registered with unique identifier
defaultSP"
}
```

The error message resembles the following code:

```
401 "error": "Insufficient user permission for role : rolename"
400 "error": "schema error:schema validation error followed by error"
400 "error": "duplicate : Idp with protocol=saml is already created"
500 "error": "Other errors"
200 OK Response
```

SAML with LDAP dependency registration

The following example is provided for IBM W3 SAML Provider and with dependency of an IBM Tivoli Directory Server LDAP. The token_attribute_mappings might change based on the SAML provider.

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource/registration/

Command

POST

Command output format

application/json

Note:

- When the LDAP is configured, the response for userinfo/introspect endpoints are directly collected from the liberty. Therefore, the attributes provided in the **token_attribute_mappings** payload are not available in the response of userinfo/introspect endpoints.
- The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.

The sample curl command resembles the following code:

```
curl -k -X POST 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v3/auth/idsource' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
--data-raw
'{
  "name": "saml-test",
  "description": "saml bluepages with ldap",
  "protocol": "saml",
  "type": "default",
  "idp_config": {
    "token_attribute_mappings": {
      "sub": "uid",
      "given_name": "firstName",
      "family_name": "lastName",
      "groups": "blueGroups",
      "email": "email"
    },
    "idp_metadata": "<base64-encoded-idp-xml>"
  },
  "ldap_config": {
    "ldap_id": "<existing_ldap_id>"
  },
  "jit": false
}'
```

The response resembles the following code:

```
{
    "status": "success",
    "message": "Identity provider {saml-test} is successfully registered with unique identifier defaultSP"
}
```

The error message resembles the following code:

```
401 "error": "Insufficient user permission for role : rolename"
400 "error": "schema error:schema validation error followed by error"
400 "error": "duplicate : Idp with protocol=saml is already created"
500 "error": "Other errors"
200 OK Response
```

Getting SAML registration by UID

You can get the IdP registration by the following ways:

- By name
- By list
- By query

Getting IdP V3 registration by name

The following example shows how to get the details of IdP V3 registration by name:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource/defaultSP

Command

GET

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.

The sample curl command resembles the following code:

```
curl -k -X GET 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v3/auth/idsource/defaultSP' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{
    "name": "w3id-sample-saml",
    "description": "w3id-sample-saml-test",
    "protocol": "saml",
    "type": "default",
    "idp_config": {
        "token_attribute_mappings": {
            "sub": "uid",
            "given_name": "firstName",
            "family_name": "lastName",
            "groups": "blueGroups",
            "email": "emailAddress"
        },
        "idp_metadata": "<base64-encoded-idp-xml>"
    },
    "jit": true,
    "uid": "defaultSP"
}
```

The error message resembles the following code:

```
401 "error": "Insufficient user permission for role : rolename"
404 "error": "Cannot find {defaultSP}
500 "error": "Other errors"
200 OK Response
```

Getting the list of IdP V3 registration

The following example shows how to get the list of IdP V3 registration:

Note: This API returns all registered IdPs of SAML or OIDC clients through IdP V3 API.

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource

Command

GET

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.

The sample curl command resembles the following code:

```
curl -k -X GET 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v3/auth/idsource' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{
    "idp": [
        {
            "name": "w3id-sample-saml",
            "description": "w3id-sample-saml-test",
            "protocol": "saml",
            "type": "default",
            "idp_config": {
                "token_attribute_mappings": {
                    "sub": "uid",
                    "given_name": "firstName",
                    "family_name": "lastName",
                    "groups": "blueGroups",
                    "email": "emailAddress"
                },
                "idp_metadata": "<base64-encoded-idp-xml>"
            },
            "jit": true,
            "uid": "defaultSP"
        }
    ]
}
```

The error message resembles the following code:

```
401 "error": "Insufficient user permission for role : rolename"
500 "error": "Other errors"
200 OK Response
```

Getting SAML registration by query

The following example shows how to get the IdP V3 registration by query:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource

Command

GET

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.

The sample curl command resembles the following code:

```
curl -k -X GET 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v3/auth/idsource?protocol=saml'
\
--header "Authorization: Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{
    "idp": [
        {
            "name": "w3id-sample-saml",
            "description": "w3id-sample-saml-test",
            "protocol": "saml",
            "type": "default",
            "idp_config": {
                "token_attribute_mappings": {
                    "sub": "uid",
                    "given_name": "firstName",
                    "family_name": "lastName",
                    "groups": "blueGroups",
                    "email": "emailAddress"
                },
                "idp_metadata": "<base64-encoded-idp-xml>"
            },
            "jit": true,
            "uid": "defaultSP"
        }
    ]
}
```

The error message resembles the following code:

```
401 "error": "Insufficient user permission for role : rolename"
500 "error": "Other errors"
200 OK Response
```

Updating SAML registration

To update the SAML registration, you require the IdP UID. To get the IdP UID, see Get the list of IdP V3 registration. Alternatively, you can use the UID as **defaultSP** because only one SAML registration at an instance is supported for now. The following example shows how to update the IdP registration:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource

Command

PUT

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.

The sample curl command resembles the following code:

```
curl -k -X PUT 'https://cp-console.apps.mycluster.mydomain.com/ idprovider/v3/auth/idsource/defaultSP' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
--data-raw
' {
  "name": "w3id-sample-saml",
  "description": "this is plain saml testing",
  "protocol": "saml",
  "type": "default",
  "idp_config": {
    "token_attribute_mappings": {
      "sub": "uid",
      "given_name": "firstName",
      "family_name": "lastName",
      "groups": "blueGroups",
      "email": "emailAddress"
    },
    "idp_metadata": "<base64-encoded-idp-xml>"
  },
  "jit": true
}'
```

The response resembles the following code:

```
{
    "status": "success",
    "message": "{defaultSP} is Updated."
}
```

The error message resembles the following code:

```
401 "error": "Insufficient user permission for role : rolename"
400 "error": "schema validation error:followed by error"
500 "error": "Other errors"
200 OK Response
```

Deleting SAML registration

To delete the SAML registration, you require the IdP UID. To get the IdP UID, see Get the list of IdP V3 registration. Alternatively, you can use the UID as **defaultSP** because only one SAML registration at an instance is supported for now. The following example shows how to delete the IdP registration:

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

idprovider/v3/auth/idsource

Command

DELETE

Command output format

application/json

Note: The curl command contains different schema elements. To understand the use of these schema elements, see Different schema elements for IdP V3.

The sample curl command resembles the following code:

```
curl -k -X DELETE 'https://cp-console.apps.mycluster.mydomain.com/idprovider/v3/auth/idsource/defaultSP' \
--header 'Content-Type: application/json' \
--header "Authorization: Bearer $ACCESS_TOKEN"
```

The response resembles the following code:

```
{
    "status": "success",
    "message": "{defaultSP} is deleted"
}
```

The error message resembles the following code:

```
400 "error": "Insufficient user permission for role : rolename"
404 "error": "Document not found"
500 "error": "Other errors"
202 OK Response
```

Getting the host details

1. Install OC (OpenShift Container) client and connect to the OCP (OpenShift Container Platform) cluster. For more information, see Getting started with the OpenShift CLI.
2. Get the hostname from the route available in **ibm-common-service** namespace or where the foundational services is installed.

```
oc get route -n ibm-common-services cp-console -o jsonpath='{.spec.host}' && echo
```

The sample output:

```
cp-console.apps.mycluster.mydomain.com
```

Different schema elements for IdP V2

- **name**: It gives the name of the IdP that you want to register. It is a mandatory field.
- **description**: It provides a short description for the registration. It is a mandatory field.
- **type**: It shows the type of the configured OIDC IdP. For example, Google, Microsoft, IBM ISV and Okta. It is a mandatory field.
- **idp_type**: It shows the type of the configured IdP. It is a mandatory field. Set the **idp_type** as **default** when you want to use the SAML with LDAP dependency. If the **idp_type** is **default**, the **saml_ldap** valid attribute is mandatory (which can be none or valid LDAP type). Currently, the supported values for the **idp_type** are **isv** or **default**.
- **saml_ldap**: It specifies the type of LDAP that you want to bind with the SAML that is available in your cluster as the IdP configuration. The following are the accepted values for the **saml_ldap** elements:
    - IBM Tivoli Directory Server
    - IBM Lotus Domino
    - IBM SecureWay Directory Server
    - Novell EDirectory
    - Sun Java System Directory Server
    - Netscape Directory Server
    - Microsoft Active Directory
    - Custom
    - None

The values of the **saml_ldap** elements are case-sensitive.

- **scim**: It is an optional field. The value of the **scim** element is **yes**. If your IdP supports the SCIM protocol, the IAM can interact with your backend SCIM server with the entries like **scim_base_path** , **scim_attribute_mappings**, and **config**. These entries are mandatory if the **scim** is set to **yes**.
- **scim_base_path**: It is the SCIM server endpoint of the IdP. It is required for uniquely identifying the service instance. It is a Mandatory field if the SCIM is set to **yes**.
- **config** elements: It is a mandatory field if the **scim** is set to **yes**.
    - **grant_type** - It identifies the token generation flow. The **client_credentials** is used to issue the tokens to access the SCIM APIs.
    - **token_url** - It is the token endpoint of the IdP. It is used to request the token.

- **client_id** - It is used for the client credentials flow.
- **client_secret** - It is used for the client credentials flow.

- **scim_attribute_mappings** - It is used to map the SCIM provider with the SCIM attributes of foundational services. You need to replace only the **values_to_be_mapped** based on your backend SCIM IdP configuration. It is a mandatory field if the **scim** is set to **yes**. The mapped values are case-sensitive.

For example,

```
"scim_attribute_mappings": {
    "user": {
        "principalName": "values_to_be_mapped”,
        "givenName": "values_to_be_mapped",
        "middleName": "values_to_be_mapped",
        "familyName": "values_to_be_mapped",
        "formatted": "values_to_be_mapped"
        },
        "group": {
            "principalName": "values_to_be_mapped",
            "created": "values_to_be_mapped",
            "lastModified": "values_to_be_mapped"
            }
        }
```

Tested values for ISV SCIM:

```
"scim_attribute_mappings": {
    "user": {
        "principalName": "userName",
        "givenName": "name.givenName",
        "middleName": "name.middleName",
        "familyName": "name.familyName",
        "formatted": "name.formatted"
        },
        "group": {
            "principalName": "displayName",
            "created": "meta.created",
            "lastModified": "meta.lastModified"
            }
     }
```

- **jit**: It is an optional field. It is **true** only if you are using the SAML without any LDAP and SCIM dependency. Otherwise, The value of **jit** is **false**.

- **token_attribute_mappings**: It is an optional field. Use the default mapping if the **token_attribute_mappings** values are not provided as the following:

```
"token_attribute_mappings": {
        "uid": "values_to_be_mapped",
        "first_name": "values_to_be_mapped",
        "last_name": "values_to_be_mapped",
        "groups": "values_to_be_mapped",
        "email": "values_to_be_mapped"
        }
```

The mapping is the IAM token claims. The mapping value is the identity provider (for example, ISV) user attribute name. If the names match, it is optional to list out the mapping, such as **given_name**, **family_name**, and **email**. The mapping shows what user attribute values to use when foundational services IAM generates ID token. The values are stored in the local database. IBM supports OIDC standard claims, such as **given_name**, **family_name**, **email** and **sub**, and additional claim groups in the mapping. You need to replace only the **values_to_be_mapped** based on the SAML configuration. The mapped values are case-sensitive.

- **status** - Its default value is **enabled**.

Different schema elements for IdP V3

- **name**: It gives the name of the IdP that you want to register. It is a mandatory field.
- **description**: It provides a short description for the registration. It is a mandatory field.
- **protocol**: It is the type of protocol. As of now, OIDC and SAML are the only supported protocols. It is a mandatory field.
- **type**: It shows the type of the configured OIDC or SAML IdP. For example, Google, Microsoft, IBM ISV and Okta for OIDC; **isv** and **default** for SAML. It is a mandatory field.
- **idp_config**: It is a mandatory field that contains the following sud-fields:

For OIDC:

- `discovery_url`: It is a well-known endpoint by OIDC provider.
- `client_id`: It is a client ID that is provided by the registered application in the IdP.
- `client_secret`: It is a client secret that is provided by the registered application in the IdP.

For SAML:

- `idp_metadata`: It is a base64 encoded value of IdP metadata XML file.
- `token_attribute_mappings`: It is used to map the token attributes with the IdP user attributes. You need to replace only the `values_to_be_mapped` based on your IDP configuration at the backend.

```
"token_attribute_mappings": {
        "groups": "values_to_be_mapped",
        "given_name": "values_to_be_mapped",
        "family_name": "values_to_be_mapped",
        "sub": "values_to_be_mapped",
        "email": "values_to_be_mapped"
        }
```

The mapped values are case-sensitive.

For SAML with SCIM dependency:

- `scim_config`: It is used to provide SCIM configuration and SCIM attribute mappings information.
    - `grant_type` - It identifies the token generation flow. The client_credentials is used to issue the tokens to access the SCIM APIs.
    - `token_url` - It is the token endpoint of the IdP. It is used to request the token.
    - `client_id` - It is used for the client credentials flow.
    - `client_secret` - It is used for the client credentials flow.

For example,

```` ``` {: pre} ````

"scim_config": { "scim_base_path": "https://bedrock-iam.verify.ibm.com/v2.0/", "grant_type": "client_credentials", "token_url": "https://bedrock-iam.verify.ibm.com/v1.0/endpoint/default/token", "client_id": "Your_isv_client_id", "client_secret": "Your_isv_client_secret", "scim_attribute_mappings": { "user": { "principalName": "userName", "givenName": "name.givenName", "middleName": "name.middleName", "familyName": "name.familyName", "formatted": "name.formatted" }, "group": { "principalName": "displayName", "created": "meta.created", "lastModified": "meta.lastModified" } } }

```
**For SAML with LDAP dependency:**

- `ldap_config`: It is used to provide LDAP dependency information.

For example,

``` {: pre}
"ldap_config":
{
"ldap_id": "existing-ldap-connection-name"
}
```

Service health and version check APIs

APIs to check the service health and API version.

Identity manager service

`icp-identity-manager` service APIs.

Swagger JSON

API version
1.0.0
API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/explorer/swagger.json

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET "https://<cluster_address>/idmgmt/explorer/swagger.json"
```

The response resembles the following code:

```
Response:
{"swagger":"2.0","info":{"version":"1.0.0","title":"platform-identity-mgmt","description":"platform-identity-
mgmt description"},\
"basePath":"/idmgmt/identity/api/v1","paths":{"/directory/ldap/{id}":{"get":{"tags":["Directory"],"summary"
...
...
{"type":"string","format":"date-time"},"expiry":{"type":"string","format":"date-time"}},"required":["id"],\
"additionalProperties":false},"ObjectID":{"type":"string","pattern":"^[a-fA-F\\d]{24}$"}}}
```

Version check

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/idmgmt/identity/api/v1/

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET "https://<cluster_address>/idmgmt/identity/api/v1/"  -I
```

The response resembles the following code:

```
Response:
HTTP/1.1 204 No Content
Server: openresty/1.13.6.2
Date: Fri, 05 Apr 2019 07:14:06 GMT
Connection: keep-alive
Vary: Origin
```

```
Access-Control-Allow-Credentials: true
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-Download-Options: noopen
X-Content-Type-Options: nosniff
X-DNS-Prefetch-Control: off
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
```

IAM policy administration service

iam-policy-administration service APIs.

Swagger JSON

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/iam-pap/v1/api-docs

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET "https://<cluster_address>/iam-pap/v1/api-docs"
```

The response resembles the following code:

```
Response:
{
  "swagger": "2.0",
  "info": {
  ...
  ...
}
```

Health check and readiness probe

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/iam-pap/v1/health

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET "https://<cluster_address>/iam-pap/v1/health"
```

The response resembles the following code:

```
Response:
OK
```

Version check

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/iam-pap/v1

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET "https://<cluster_address>/iam-pap/v1"  -I
```

The response resembles the following code:

```
Response:
HTTP/1.1 204 No Content
Server: openresty/1.13.6.2
Date: Fri, 05 Apr 2019 07:24:53 GMT
Connection: keep-alive
transaction-id: e4574842-52e0-4734-a7e4-5e13c2099ad3
X-Response-Time: 0.708ms
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
```

IAM token service

iam-token-service service APIs.

Version check

API version

1.0.0

API URI components

Scheme

HTTPS

Host IP

Cluster Master Host

Port number

Cluster Master API Port

Path

/iam-token/v1

Command

GET

Command output format

application/json

The sample curl command resembles the following code:

```
curl -k -X GET "https://<cluster_address>/iam-token/v1" -I
```

The response resembles the following code:

```
Response:
HTTP/1.1 204 No Content
Date: Wed, 08 May 2019 06:59:50 GMT
Content-Type: text/plain
Content-Length: 0
Connection: keep-alive
X-Powered-By: Servlet/3.1
Content-Language: en-US
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
```

APIs

Licensing APIs are available to help you retrieve license usage data per cluster from License Service and for multiple clusters from License Service Reporter.

- Retrieving license usage data per cluster from License Service
- Retrieving license usage data for multiple clusters from License Service Reporter

Retrieving license usage data per cluster from License Service

You can use License Service to retrieve license usage for a single cluster through API calls.

License Service collects, measures, and stores the license usage data for a cluster. You can use dedicated API calls to retrieve this data.

- API authentication
- API calls for retrieving license usage data
- Obtaining a status page
- License Service Swagger API schema

API authentication

To retrieve License Service data with the dedicated APIs, you need to use the proper authentication method. You can authenticate by using the service account or License Service token.

Authentication methods:

- License Service API token - Basic authentication method that uses a static token which is generated and saved in the License Service secret. This token is passed as the following URL parameter: **`?token=<token>`**.

If your security policy does not allow you to provide a token as URL parameter, you can disable this option. For more information, see Disabling License Service API token as URL parameter.

- Service account token - Advanced and customizable authorization method that is based on a service account token which is provided by the request header field.

Obtaining and updating an API token

Obtain the License Service API token to authenticate and retrieve license usage data. Learn how to update the token to comply with your security policies if needed.

License Service API token is a basic authentication method for connecting with License Service APIs. If you require to configure advanced API authentication, see Service account token.

- Using the License Service API token
- Obtaining an API token
- Updating an API token
- Disabling the API token as URL parameter

Using the License Service API token

To connect to the API endpoint, you need to provide a secure authentication token as the URL parameter in the API URL.

The following examples show how to use a License Service token in your API URL:

Table. API URL format

API	API URL format
Retrieving an audit snapshot	`<License Service URL>/snapshot?token=<token>`
Retrieving license usage of products	`<License Service URL>/products?token=<token>`
Retrieving license usage of bundled products	`<License Service URL>/bundled_products?token=<token>`
Retrieving contribution of services	`<License Service URL>/services?token=<token>`
Retrieving information about License Service health	`<License Service URL>/health?token=<token>`
Obtaining the status page	`<License Service URL>/status?token=<token>`

Note: You do not need to use an API to retrieve information about the version of License Service that is deployed on a cluster.

Obtaining an API token

Obtaining an API token by using the CLI

Note: The commands assume that the IBM Cloud Pak foundational services are installed in `ibm-common-services` namespace. If IBM Cloud Pak for AIOps installs foundational services in a different namespace, adjust the commands.

- For Linux and OSX run the following commands.
  - **Linux**: `kubectl get secret ibm-licensing-token -o jsonpath={.data.token} -n ibm-common-services | base64 -d`
  - **OSX**: `kubectl get secret ibm-licensing-token -o jsonpath={.data.token} -n ibm-common-services | base64 -D`

  The value that you get in the response is your API token.
- For Windows run the following command.
  - **Windows**: `kubectl get secret ibm-licensing-token -o jsonpath={.data.token} -n ibm-common-services`

    The token that you retrieve is encoded. To decode the token, you can use the Certificate Services, if available. Run the following commands to retrieve and encode the token with Certificate Services:

    ```
    kubectl get secret ibm-licensing-token -o jsonpath={.data.token} -n ibm-common-services > token_encoded

    certutil -decode token_encoded token_decoded

    type token_decoded
    ```

    Otherwise, obtain the token by using the OpenShift console. Complete the steps in the following procedure.

Obtaining an API token by using the OpenShift console

1. Log in to the OpenShift console of the cluster where you deployed License Service.
2. Go to **Workloads** > **Secrets**.
3. Set the project to **All Projects**.
4. Find the **ibm-licensing-token** and select it.
5. Scroll to the **Data** section and copy the token by using the **Copy to Clipboard** option.

Updating an API token

Important: The API token cannot contain spaces or the following characters `?`,`%`, and `&`.

Updating an API token by using the CLI

Note: The commands assume that the IBM Cloud Pak foundational services are installed in `ibm-common-services` namespace. If IBM Cloud Pak for AIOps installs foundational services in a different namespace, adjust the commands.

1. Create a `token.yml` file with the following contents. Provide your new token as `<TOKEN>`.

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    app.kubernetes.io/component: ibm-licensing-service-svc
    app.kubernetes.io/instance: ibm-licensing-service
    app.kubernetes.io/managed-by: operator
    app.kubernetes.io/name: ibm-licensing-service-instance
    release: ibm-licensing-service
  name: ibm-licensing-token
  namespace: ibm-common-services
type: Opaque
stringData:
  token: <TOKEN>
```

2. To apply the new token, run the following command.

```
kubectl apply -f token.yml
```

3. To refresh the application by deleting the pod, run the following command.

```
kubectl delete pods -l app=ibm-licensing-service-instance -n ibm-common-services
```

Updating an API token by using the OpenShift console

1. Open the OpenShift console of the cluster where you deployed License Service.
2. Go to **Workloads** > **Secrets**.
3. Set the project to **All Projects**.
4. Find the **ibm-licensing-token** and select it.
5. From the **Actions** list, select **Edit Secret**.
6. Edit the secret value and click **Save**.

Disabling the API token as URL parameter

If your security policy does not allow you to provide a token as a URL parameter, you can disable this option. Then, you must use a service account token as an authentication method. Before you change the authentication method, see Service account token.

1. Edit the IBMLicensing custom resource.
2. Add the following lines to the `IBMLicensing` section, under `spec`:

```
features:
  auth:
    urlBasedEnabled: false
```

Note: To turn the License Service token authentication back on, remove these lines from the custom resource.

Service account token

Learn how to use the service account token to connect with the License Service API server.

Service account token authentication provides flexible and customizable way to manage access to License Service APIs which is based on a role-based access control (RBAC).

The following article describes basic configuration. For more advanced configuration, see Using RBAC Authorization in Kubernetes documentation.

- Limitations
- Authentication and authorization
- Default configuration
- Custom configuration
- Using the API token

Limitations

Service account token is provided via `http` header. Because of that to retrieve the licensing data via the APIs from your browser, you must use the simplified License Service interface which is available under License Service URL. For more information, see Obtaining the License Service URL.

Authentication and authorization

- **Authentication** - Checks whether the provided service account token is correct for the service account on a cluster where License Service is deployed.
- **Authorization** - Checks whether the service account has a binding to License Service API endpoints.

Default configuration

By default, after installation, the **ibm-licensing-default-reader** service account is created in the namespace where License Service is deployed.

This service account has a binding to the cluster role that gives access to License Service APIs.

Obtaining the token for `ibm-licensing-default-reader`

By using the CLI

Complete the following steps to get the service account token by using kubectl:

Note: The commands assume that the IBM Cloud Pak foundational services are installed in `ibm-common-services` namespace. If IBM Cloud Pak for AIOps installs foundational services in a different namespace, adjust the commands.

1. Get the name of your Kubernetes secret object. Secrets are used to store access credentials.

   ```
   kubectl get secret -n ibm-common-services | grep ibm-licensing-default-reader-token | head -n 1
   ```

   Note:

   - `ibm-licensing-default-reader-token-xxxxx` can be either `ibm-licensing-default-reader-token` without any hash or with some hash. For example,`ibm-licensing-default-reader-token-btjdk`.
   - If there are no tokens found, then you might have kubernetes 1.24+ and not updated License Service, you can either update License Service or just create token:

   ```
   apiVersion: v1
   kind: Secret
   type: kubernetes.io/service-account-token
     metadata:
      name: ibm-licensing-default-reader-token
      namespace: ibm-common-services
      annotations:
        kubernetes.io/service-account.name: "ibm-licensing-default-reader"
   ```

   After creating this secret if `ibm-licensing-default-reader` service account exists kubernetes should add `data.token` in this secret, containing bearer token for this service account.

2. Retrieve the **ibm-licensing-default-reader** token.

   - For Linux and OSX run the following commands.
     - **Linux:** `kubectl get secret ibm-licensing-default-reader-token-xxxxx -o jsonpath={.data.token} -n ibm-common-services | base64 -d`
     - **OSX:** `kubectl get secret ibm-licensing-default-reader-token-xxxxx -o jsonpath={.data.token} -n ibm-common-services | base64 -D`

   The value that you get in the response is your API token.

   - For Windows run the following command.

- **Windows:** `kubectl get secret ibm-licensing-default-reader-token-xxxxx -o jsonpath={.data.token} -n ibm-common-services`

Decode the returned base64 value to reveal your API token.

By using the OpenShift console

1. Log in to the OpenShift console of the cluster where you deployed License Service.
2. Go to **Workloads** > **Secrets**.
3. Find the **ibm-licensing-default-reader-token-xxxxx** secret and select it.

   Note: If more than one secret is listed, select any of these secrets.
4. Scroll to the **Data** section, and copy the token by clicking the **Copy to Clipboard** icon.

Custom configuration

You can create and use a different service account that has a binding to a proper cluster role to access License Service API. For more information, see Using RBAC Authorization in Kubernetes documentation.

Cluster role configuration

Configure a cluster role with the following content:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <role name>
rules:
  - nonResourceURLs:
      - "/products"
      - "/snapshot"
      - "/bundled_products"
      - "/services"
      - "/health"
      - "/status"
    verbs:
      - "get"
```

where

nonResourceURLs is a list of the APIs to which the service account should have access. You can customize this list to met your requirements.

The list of available **nonResourceURLs**:

Table. API nonresourceURLS

API	nonResourceURLs
Retrieving an audit snapshot	`/snapshot`
Retrieving license usage of products	`/products`
Retrieving license usage of bundled products	`/bundled_products`
Retrieving contribution of services	`/services`
Retrieving information about License Service health	`/health`
Obtaining the status page	`/status`

Important: This list might change in the future if new APIs are added. Monitor the list to make sure that you update the cluster role with access have access to all APIs.

Creating cluster role bindings

Create a cluster role binding for your custom service account with the following content:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: <binding name>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: <custom role name>
subjects:
  - kind: ServiceAccount
    name: <service account name>
    namespace: <service account namespace>
```

Creating service account token if needed

If you have kubernetes 1.24 or higher, then after creating your service account, the secret with bearer token would not be created by default. You need to create it manually:

```
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: <your_service_account_name>-token
  namespace: <namespace_where_you_have_your_service_account>
  annotations:
    kubernetes.io/service-account.name: "<your_service_account_name>"
```

Adding such annotations will make Kubernetes add token inside this secret in `data.token`. Then, you can retrieve your token in the same way as described in Step 2 of Obtaining token. But change the secret name and namespace to where you have it:

For Linux and OSX, run the following commands:

Linux:

```
kubectl get secret <your_service_account_name>-token -o jsonpath={.data.token} -n
<namespace_where_you_have_your_service_account> | base64 -d
```

OSX:

```
kubectl get secret <your_service_account_name>-token -o jsonpath={.data.token} -n
<namespace_where_you_have_your_service_account> | base64 -D
```

The value that you get in the response is your API token.

For Windows, run the following command:

```
kubectl get secret <your_service_account_name>-token -o jsonpath={.data.token} -n
<namespace_where_you_have_your_service_account>
```

Decode the returned base64 value to reveal your API token.

Using the service account token

To connect to the API endpoint, you need to provide the secure authentication token. If you choose to use a service account token, you need to add it to the header for every API call.

The following example shows how to provide the service account token to access the License Service API.

```
curl -k -H "Authorization:Bearer <token>" <API URL>
```

The list of License Service APIs:

Table. API URLs

API	API URL
Retrieving an audit snapshot	`<License Service URL>/snapshot`
Retrieving license usage of products	`<License Service URL>/products`
Retrieving license usage of bundled products	`<License Service URL>/bundled_products`
Retrieving contribution of services	`<License Service URL>/services`
Retrieving information about License Service health	`<License Service URL>/health`
Obtaining the status page	`<License Service URL>/status`

APIs for retrieving License Service data

You can retrieve the aggregated license usage data for a cluster from the last 24 months by using dedicated API calls. Additionally, you can use the APIs to retrieve License Service version and health.

Available APIs

Table. Available Api descriptions

API	Description
Retrieving an audit snapshot	Retrieve an audit snapshot that is needed to fulfill the requirements of Container Licensing rules.

API	Description
Retrieving license usage of products	Retrieve the list of products that are deployed on the cluster with their license usage. Use this API to integrate with external tools to monitor you license usage.
Retrieving license usage of bundled products	Retrieve the list of bundled products that are deployed on the cluster with their license usage. Use this API to integrate with external tools to monitor you license usage.
Retrieving contribution of services	Retrieve the contribution of services in the overall license usage of your bundled products. This information is additional and applies only to limited number of enabled products.
Retrieving information about License Service version	Retrieve information about License Service version for troubleshooting or upgrade purposes.
Retrieving information about License Service health	Retrieve information about License Service health to identify problems.

For an overview of all available APIs, see: License Service API schema.

Obtaining the License Service URL

To obtain the License Service URL that is required to retrieve license usage data from the cluster, complete the following steps:

1. Open the OpenShift console and go to **Networking** > **Routes**.
2. Set the project to **All Projects**.
3. Find the **ibm-licensing-service-instance** route and copy the URL listed as **Location**. It is your License Service URL.

Note: Alternatively, you can use the following command to obtain the License Service URL: `kubectl get routes -n ibm-common-services | grep ibm-licensing-service-instance | awk '{print $2}'`. This command assumes that the IBM Cloud Pak foundational services are installed in `ibm-common-services` namespace. If IBM Cloud Pak for AIOps installs foundational services in a different namespace, adjust the command.

You can use the License Service URL to access the simplified License Service interface for testing purposes. Paste the URL into your browser, select one of the APIs, and provide the authorization token. The interface is limited and retrieves the licensing data for last 30 days for all products.

API authentication

To use the License Service APIs, you need to authenticate first. To learn how to get the secure authentication token, see API authentication.

Retention period

You can use the APIs to retrieve the aggregated license usage for the last 24 months.

Retrieving an audit snapshot

API URL

`<License Service URL>/snapshot`

API parameters

- Optional: `startDate`, `endDate`, `metricName`
- Required: If you use the default authentication method, the License Service token, you need to provide the `token` parameter in the API URL. For more information about how to retrieve the token, see Obtaining the License Service token.
- Example: Retrieving an audit snapshot for the specific period

  To retrieve an audit snapshot for the specific period, define the `startDate` and `endDate` parameters. The date should have the following format YYYY-MM-DD.

  - Using the License Service token:

    API URL:

    `<License Service URL>/snapshot?token=<token>&startDate=<YYYY-MM-DD>&endDate=<YYYY-MM-DD>`

    For example:

    ```
    https://ibm-licensing-service-instance-ibm-common-services.apps.hostname.example/snapshot?
    token=YourToken&startDate=2020-09-01&endDate=2020-09-30
    ```

  - Using the Service account token:

    API URL:

    `<License Service URL>/snapshot?startDate=<YYYY-MM-DD>&endDate=<YYYY-MM-DD>`

For example:

```
https://ibm-licensing-service-instance-ibm-common-services.apps.hostname.example/snapshot?
startDate=2020-09-01&endDate=2020-09-30
```

For more information about the API parameters and how to use them, see License Service API schema.

API description

This API call retrieves a license usage snapshot for your cluster. An audit snapshot is a compressed `.zip` package that includes a complete set of audit documents that certify your license usage per cluster.

To view a list of files that are included in an audit snapshot, see Content of the audit snapshot.

Integrating with the audit snapshot

To integrate with the audit snapshot, use the names of the columns from the `.csv` files as the point of reference, not the column order. The names of the columns are fixed. However, the order of the columns might change.

Retrieving license usage of products

API URL

```
<License Service URL>/products
```

API parameters

- Optional: `startDate`, `endDate`, `format`, `metricName`, `groupName`
- Required: If you use the default authentication method, the License Service token, you need to provide the `token` parameter in the API URL. For more information about how to retrieve the token, see Obtaining the License Service token.

For more information about the API parameters and how to use them, see License Service API schema.

API description

- This API call retrieves information about the products that are deployed on a cluster with the value and date of the license usage peak. By default, the retrieved data includes information about IBM Cloud Pak for AIOps, all IBM Cloud Paks, or stand-alone containerized products that are deployed on a cluster.
- This API call retrieves information about the products that are deployed in namespaces that belong to user-defined groups. To filter the data by group use the groupName parameter: `<License Service URL>/products?groupName=<groupName>`. To enable this feature, see Filtering the license usage data by user-defined groups.

This API retrieves the following information per product:

Table. Product license usage descriptions

Information	Description
name	The name of the product.
id	The identifier of the product.
metricPeakDate	The date when the license metric usage of the product was the highest within the requested period.
metricQuantity	The highest number of license units that the product used within the requested period.
metricName	The license metric unit that is used by the product.

Retrieving license usage of bundled products

API URL

```
<License Service URL>/bundled_products
```

API parameters

- Optional: `startDate`, `endDate`, `format`, `metricName`, `groupName`
- Required: If you use the default authentication method, the License Service token, you need to provide the `token` parameter in the API URL. For more information about how to retrieve the token, see Obtaining the License Service token.

For more information about the API parameters and how to use them, see License Service API schema.

API description

This API call retrieves information about the bundled products that are included in IBM Cloud Paks that are deployed on a cluster with the highest license usage within the requested period.

This API retrieves the following information per each bundled product:

Table. Bundled products license usage information

	Description
productName	The name of the detected bundled product.
productId	The identifier of the bundled product.
metricName	The license metric unit that is used by the bundled product.
metricPeakDate	The date when the license usage of the bundled product was the highest within the requested period.
metricMeasuredQuantity	The highest number of license units that the bundled product used within the requested period.
cloudpakId	The identification number of the IBM Cloud Pak® to which the program is bundled.
cloudpakVersion	Version of the IBM Cloud Pak® to which the program is bundled.
cloudpakMetricName	The license metric unit that is used by the entire IBM Cloud Pak® to which the bundled product contributes.
metricConversion	The ratio that shows how the license usage of the bundled product is counted when compared with the license usage of the IBM Cloud Pak®. It shows how the program's license metrics are recalculated when compared to the IBM Cloud Pak® license metrics.
metricConvertedQuantity	The number of license units that the bundled product contributed to the overall license usage of the IBM Cloud Pak®. The value is calculated by comparing metricMeasuredQuantity against metricConversion.

Retrieving contribution of services

This API is available from IBM Cloud Pak foundational services version 3.19.x.

API URL

```
<License Service URL>/services
```

API parameters

- Optional: `startDate`, `endDate`, `format`, `metricName`
- Required: If you use the default authentication method, the License Service token, you need to provide the `token` parameter in the API URL. For more information about how to retrieve the token, see Obtaining the License Service token.

For more information about the API parameters and how to use them, see License Service API schema.

API description

This API call retrieves information about how the usage of services contributes to the overall license usage of the bundled products under which these services are grouped. This information might give you a better understanding of your license usage, however, it is not obligatory from the Container Licensing rules perspective.

You can use this API to retrieve information only for the products that have three-layer reporting structure that includes services that are grouped under bundled products, and are enabled for reporting, such as IBM Cloud Pak for Data.

This API retrieves the following information:

Table. Contribution of services information

Information	Description
cloudpakId	The identifier of the Cloud Pak with which the bundled product and its services are deployed.
cloudpakMetricName	The license metric unit that is used by the product.
productName	The name of the bundled product.
productId	The identifier of the bundled product.
metricName	The license metric unit that is used by the bundled product.
serviceName	The name of the service that is deployed under the bundled product.
serviceId	The identifier of the service.
serviceMetricValue	The highest number of license units that the service used that contributed to the overall license usage of its bundled product.
metricDate	The date when the license metric usage of the product was the highest within the requested period.

Retrieving information about License Service version

API URL

```
<License Service URL>/version
```

API description

This API call retrieves the following information about the License Service version:

Table. License Service version information

	Description
version	The version of License Service that is deployed on a cluster.
buildDate	The buildDate identifies the image that was used to deploy License Service and might be needed for support purposes.
commit	The commit ID identifies the image that was used to deploy License Service and might be needed for support purposes.

Retrieving information about License Service health

API URL

```
<License Service URL>/health
```

API parameters

- Required: If you use the default authentication method, the License Service token, you need to provide the `token` parameter in the API URL. For more information about how to retrieve the token, see Obtaining the License Service token.

API description

This API call retrieves the following information:

Table. License Service health information

	Description
count	The number of pods that are deployed on the cluster but have incomplete or improper product annotations. Improper annotations result in the license usage of a product not being measured. The annotations include the product metadata, such as the product ID and metric, and are implemented by the IBM team that develops the product to enable license usage measurements.
pods	The list of pods that that have improper annotations.

Obtaining status page to view the license usage details

Learn how to obtain the status page that contains the most important information about your deployments and their license usage. You can use this data for analysis or troubleshooting.

This feature is available from IBM Cloud Pak® foundational services version 3.7.x.

The status page is an html page that summarizes the most important information about your deployments on the cluster. The status page always shows the information that are collected and valid at the moment of page retrieval.

Retrieving the status page

To retrieve the status page, complete the following actions:

1. Obtain the License Service URL.
2. Get the authentication token.
3. Open the License Service URL in your browser.
4. Select the status page.
5. Provide your authentication token.

Status page details

The status page lists the following information in dedicated sections:

- The current information about how the overall license usage of the Virtual Processor Core (VPC) and Processor Value Unit (PVC) metrics is calculated for the cluster.
- License usage of products and bundled products that are deployed on a cluster.
- The list of contributing pods per node.

- The nodes and pods that are running on the cluster.
- Information about capping.
- Information about pod CPU limit and node CPU capacity.
- Pod annotations that allow the product license usage to be calculated by License Service.
- Information about the namespace where the pod is deployed.

You can generate the html status page on demand to understand how the license usage of your products is calculated and make sure that the product deployments are correct and optimal for you use.

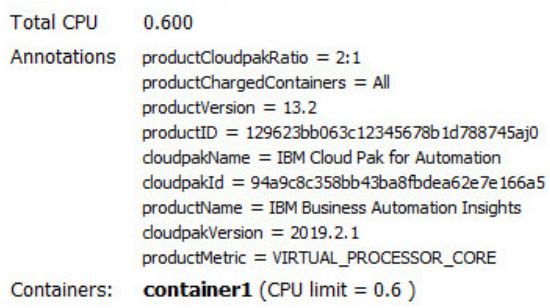


Figure. Example HTML status page

Retrieving license usage data for multiple clusters from License Service Reporter

You can use License Service Reporter to retrieve license usage for multiple clusters through API calls.

License Service Reporter aggregates the license usage data from multiple clusters that are configured to deliver the license usage data. You can use dedicated API calls to retrieve the collective data.

- Obtaining and updating the API token
- APIs for retrieving License Service Reporter data
- License Service Reporter Swagger API schema

Obtaining and updating an API token

Obtain an API token that is required to make the API calls and retrieve license usage data from License Service Reporter. Learn how to update the token to comply with your security policies if needed.

Note: License Service Reporter is supported on OpenShift, and the best practice is to obtain and update the token by using the OpenShift console.

Obtaining an API token

Obtaining an API token by using the OpenShift console

1. Log in to the OpenShift console of the cluster where you deployed License Service Reporter.
2. Go to **Workloads** > **Secrets**.
3. Set the project to **All Projects**.
4. Find the **ibm-licensing-reporter-token** and select it.
5. Scroll to the **Data** section and copy the token by using the **Copy to Clipboard** option.

Obtaining an API token by using the CLI

- For Linux and OSX run the following commands.
  - **Linux**: `oc get secret ibm-licensing-reporter-token -o jsonpath={.data.token} -n ibm-common-services | base64 -d`
  - **OSX**: `oc get secret ibm-licensing-reporter-token -o jsonpath={.data.token} -n ibm-common-services | base64 -D`

  The value that you get in the response is your API token.
- For Windows run the following command.

  Windows: `oc get secret ibm-licensing-reporter-token -o jsonpath={.data.token} -n ibm-common-services`

  Decode the returned base64 value to reveal your API token.

Updating an API token

Important: The API token cannot contain spaces or the following characters `?`,`%`, and `&`.

Updating an API token by using the OpenShift console

1. Open the OpenShift console of the cluster where you deployed License Service Reporter.
2. Go to **Workloads** > **Secrets**.
3. Set the project to **All Projects**.
4. Find the **ibm-licensing-reporter-token** and select it.
5. From the **Actions** list, select **Edit Secret**.
6. Edit the secret value and click **Save**.
7. Restart the License Service pod.

Updating an API token by using the CLI

1. Create a `token.yml` file with the following contents. Provide your new token as `<TOKEN>`.

```
apiVersion: v1
kind: Secret
metadata:
  labels:
    app.kubernetes.io/component: ibm-licensing-service-svc
    app.kubernetes.io/instance: ibm-licensing-service
    app.kubernetes.io/managed-by: operator
    app.kubernetes.io/name: ibm-licensing-service-instance
    release: ibm-licensing-service
  name: ibm-licensing-reporter-token
  namespace: ibm-common-services
type: Opaque
stringData:
  token: <TOKEN>
```

2. To apply the new token, run the following command.

```
oc apply -f token.yml
```

3. To refresh the application by deleting the pod, run the following command.

```
oc delete pods -l app=ibm-license-service-reporter-instance -n ibm-common-services
```

4. Restart the License Service pod.

API calls for retrieving License Service Reporter data

You can use the dedicated API calls to retrieve information about the aggregated license usage data for your multicluster environment.

Available APIs

- Retrieving an audit snapshot
- Retrieving license usage of products
- Retrieving license usage of bundled products
- Retrieving information about data sources
- Retrieving information about License Service Reporter version

Obtaining License Service Reporter URL

To obtain the License Service Reporter URL that is required to run the APIs, complete the following steps.

1. Open the OpenShift console of the cluster where you deployed License Service Reporter.
2. Go to **Networking** > **Routes**.
3. Set the project to **ibm-common-services**.
4. Find the **ibm-license-service-reporter** route and copy the URL listed as **Location**. It is your License Service Reporter URL.

Note: Alternatively, you can use the following command to obtain the License Service Reporter URL: `oc get routes -n ibm-common-services | grep ibm-license-service-reporter | awk '{print $2}'`

Obtaining an API token

To learn how to get the authorization token, see Obtaining and updating an API token.

Retrieving an audit snapshot

API URL

`<License Service Reporter URL>/snapshot?token=<token>`

API parameters

- Required: `token`
- Optional: `startDate`, `endDate`, `metricName`

For more information about the API parameters and how to use them, see License Service Reporter Swagger API schema.

API description

This API call retrieves a snapshot of license usage for your multicluster environment.

This APIs returns the .zip package that contains the collection of audit snapshot packages for each data source. Each audit snapshot is a compressed `.zip` package that includes a complete set of audit documents that certify your cumulative license usage. By default, this API retrieves the audit snapshot for the last 30 days.

Note: The files contain information about the highest license usage basing on the data from multiple clusters. The highest license usage is expressed in the `metricQuantity` column. For more information, see the definition of the highest license usage.

To view a list of files that are included in each audit snapshot package, see Content of the audit snapshot.

Integrating with the audit snapshot

To integrate with the audit snapshot, use the names of the columns from the `.csv` files as the point of reference, not the column order. The names of the columns are fixed. However, the order of the columns might change.

Retrieving license usage of products

API URL

`<License Service Reporter URL>/products?token=<token>`

API parameters

- Required: `token`
- Optional: `startDate`, `endDate`, `format`, `metricName`

For more information about the API parameters and how to use them, see License Service Reporter Swagger API schema.

API description

This API call retrieves information about the products that are deployed on the clusters that deliver data to License Service Reporter. The API returns information about the value and date of the license usage peak. By default, the retrieved data includes information about all IBM Cloud Paks and stand-alone containerized products that are deployed in the multicluster environment for the last 30 days.

This API retrieves the following information per product:

Table. Product license usage information

Information	Description
name	The name of the product.
id	The identifier of the product.
metricPeakDate	The date and time when the license usage of the product on all clusters was the highest within the requested period.
metricQuantity	The highest license usage that is expressed in the number of license units that the product used on a `metricPeakDate`.
metricName	The license metric unit that is used by the product.

Retrieving license usage of bundled products

API URL

`<License Service Reporter URL>/bundled_products?token=<token>`

API parameters

- Required: `token`
- Optional: `startDate`, `endDate`, `format`, `metricName`

For more information about the API parameters and how to use them, see License Service Reporter Swagger API schema.

API description

This API call retrieves information about the highest license usage within the requested period of the bundled products that are deployed on clusters as parts of IBM Cloud Paks. By default, this API retrieves the license usage for the last 30 days.

This API retrieves the following information per each bundled product.

Table. Bundled porduct license information

Information	Description
productName	The name of the detected bundled product.
productId	The identifier of the bundled product.
metricName	The license metric unit that is used by the bundled product.
metricPeakDate	The date and time when the license usage of the bundled product on all clusters was the highest within the requested period.
metricMeasuredQuantity	The highest license usage that is expressed in the number of license units that the product used on a `metricPeakDate`
cloudpakId	The identification number of the IBM Cloud Pak® to which the program is bundled.
cloudpakVersion	Version of the IBM Cloud Pak® to which the program is bundled.
metricConversion	The ratio that shows how the license usage of the bundled product is counted when compared with the license usage of the IBM Cloud Pak®. It shows how the license metrics of a bundled product are recalculated when compared to the IBM Cloud Pak® license metrics.
cloudpakMetricName	The license metric unit that is used by the entire IBM Cloud Pak® to which the bundled product contributes.
metricConvertedQuantity	The number of license units that the bundled product contributed to the overall license usage of the IBM Cloud Pak®. The value is calculated by comparing `metricMeasuredQuantity` against `metricConversion`.

Retrieving information about data sources

API URL

```
<License Service Reporter URL>/datasources?token=<token>
```

API parameters

- Required: `token`

For more information about the API parameters and how to use them, see License Service Reporter Swagger API schema.

API description

This API call retrieves information about the clusters that deliver license usage data to License Service Reporter.

This API retrieves the following information per product.

Table. Data source information

Information	Description
datasourceId	The unique identifier of the cluster.
datasourceName	The name of the cluster as it is displayed on the License Service Reporter user interface.
datasourceType	The type of the data source. • `LS` - License Service. • `LMT` - License Metric Tool
lastReported	The date and time when the license usage data was last delivered from the data source to License Service Reporter.

Retrieving information about License Service Reporter version

API URL

```
<License Service Reporter URL>/version
```

API description

This API call retrieves the following information about the License Service Reporter version:

Table. Licence Service Reporter version information

Information	Description
version	The version of License Service Reporter that is deployed on a cluster.

Information	Description
buildDate	The buildDate identifies the image that was used to deploy License Service Reporter and might be needed for support purposes.
commit	The commit ID identifies the image that was used to deploy License Service Reporter and might be needed for support purposes.

Issue resolution API

The issue resolution API allows for the interaction with alerts, incidents and related artifacts.

Endpoint URLs

Issue resolution events API

```
cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/issue-resolution/v1/events/
```

Issue resolution alerts API

```
cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/issue-resolution/v1/alerts/
```

Issue resolution incidents API

```
cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/issue-resolution/v1/incidents
```

Authentication

The API uses Platform UI (Zen) authentication, so you either need a JWT token or ZenApiKey to be authenticated. The token or key is used to determine whether a user or service ID has access when you use the API.

Enabling access to the Probable cause customization APIs

The Probable cause API ranks the alerts in an incident. This API uses a classifier that uses Natural Language Processing (NLP) to determine the meaning of an alert and calculates the topology path between alerts to create a ranking for each applicable alert.

You can use the Probable cause API to customize the analytics for probable cause, and add edge configurations when new custom edge types are added to IBM Cloud Pak for AIOps.

The Probable cause customization APIs are exposed by default, but to access them you must generate the token first. The API uses the IBM Cloud Pak Platform UI authentication, so you need a JWT (JSON Web Token) to be authenticated. The token is used to determine whether a user or service ID has access when they use the API.

Probable Cause Customization APIs

1. Log in to the cluster.
2. Set ROUTE to **cpd** route.

   For example:

   ```
   ROUTE=$(oc get route cpd -n cp4aiops --no-headers | awk '{print $2}')
   ```

3. Use the admin-user-details secret to set PASS.

   ```
   PASS=$(oc get secret admin-user-details -o jsonpath='{.data.initial_admin_password}' -n cp4aiops | base64 -d)
   ```

4. Create the TOKEN.

   ```
   TOKEN=$(curl -s -k -X POST https://$ROUTE/icp4d-api/v1/authorize -H 'Content-Type: application/json' -d '{"username": "admin","password": "password"}' | jq .token | sed 's/"//g')
   ```

Example of accessing the APIs

You can now access the APIs, for example:

`/customisation/edge_type`

Use a GET request `/customisation/edge_type` to get the list of edges in the system.

```
curl -k -X GET --header 'Accept: application/json' -H "Authorization: Bearer ${TOKEN}"
https://$ROUTE/aiops/api/issue-resolution/mime/v1/customisation/edge_type -H "accept: application/json" -H
"X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255"
```

The label and edge names are the same as those added in the topological resource group. Some edges have different weight due to directorial. Overall, an `indegree` edge has a higher weight than an `outdegree` edge.

Note: If the system encounters an `unknown` edge type, then it uses the average of weight in the edge label group.

The following example shows a response(200) for the GET request:

```
{
"edge_type_list": [
   {
      "edge": "string",
      "label": "AGGREGATION",
      "outdegree": 0,
      "indegree": 0
   }
]
}
```

Use a POST request `/customisation/edge_type` to overwrite the complete list of edge types.

```
curl -k -X POST -H "Content-Type: application/json" -H "Authorization: Bearer ${TOKEN}"
https://$ROUTE/aiops/api/issue-resolution/mime/v1/customisation/edge_type -H "X-TenantID: cfd95b7e-3bc7-4006-
a4a8-a73a79c71255" -d '{"edge_type_list":[{ "edge": "newEdgeName", "label": "association", "outdegree":
20,"indegree": 5}]}'
```

The following example shows a response(200) for the POST request:

```
{"Result":"Success"}
```

After the path score is calculated, the summary is classified in one of the following 8 categories.

Table 1. Label and Weight relationship for classification of summary

Label	Weight
errorrate	1.4
exception	1.2
information	1
latency	2
saturation	2.2
stateChange	1.8
traffic	2.4
unknown	-1

In order to modify the label weight, use the API `/customisation/label_weight`.

`/customisation/label_weight`

Use a POST request `/customisation/label_weight` to add or update label weight. The request body is an array of label weight objects.

```
curl -k -X POST -H "Content-Type: application/json" -H "Authorization: Bearer ${TOKEN}"
https://$ROUTE/aiops/api/issue-resolution/mime/v1/customisation/label_weight -H "X-TenantID: cfd95b7e-3bc7-
4006-a4a8-a73a79c71255" -d '{"label_weight_list":[{"label":"association","weight":1.2}]}'
```

The following example shows a response(200) for the POST request:

```
{"Result":"Success"}
```

Note: Before removing a label from the list, remove all of the training examples from the classifier. The classifier looks at the word in the summary and attempts to put the alert in the appropriate category.

`/customisation/words`

Use a GET request `/customisation/words` to get the list of words in the alert summary that is used by probable cause for the specified tenant ID. If the GET request is successful(200 response), the response includes an array of `words` for the alert.

```
curl -k -X GET --header 'Accept: application/json' -H "Authorization: Bearer ${TOKEN}"
https://$ROUTE/aiops/api/issue-resolution/mime/v1/customisation/words -H "accept: application/json" -H "X-
TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255"
```

The following example shows a response(200) for the GET request:

```
{
   "words":[
      {
         "word":"”error”",
         "”caseSenstive”":false,
         "weight":100
      },
      {
         "word":"”ECC”",
         "”caseSenstive”":true,
         "weight":10
      }
   ]
}
```

If both words "error" and "ECC" are found in the same alert summary, then the weight(score) is added. For example, if **error** has a weight of 100 and **ECC** has a weight of 10, then the total word score is 110. The higher the score, the more likely that this alert is a root cause.

Use a POST request **/customisation/words** to update(replace) the words in the system. The request body is the JSON array of words for probable cause.

```
curl -k -X POST --header 'Accept: application/json' -H "Authorization: Bearer ${TOKEN}" -H "Content-Type:
application/json" https://$ROUTE/aiops/api/issue-resolution/mime/v1/customisation/words -H "accept:
application/json" -H "X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255" -d '{"words":
[{"word":"error","caseSenstive": false, "weight":100}, {"word":"ECC","caseSenstive": true, "weight":10}]}'
```

The following example shows a response(200) for the POST request:

```
{"Result":"Success"}
```

/customisation/severity

Use a GET request **customisation/severity** to get the list of severities that is used by probable cause for the specified tenant ID. Based on the severity of the alert, the weight is added to the word score. Refer to the following table for more details:

Table 2. Severity and Weight relationship

Severity	Weight
1	10
2	20
3	30
4	40
5	50
6	60

If the word score is 110(Error and ECC from previous example) and the severity of the alert is critical(which is 6), then 60 is added to the score, which results in a word score of 170.

```
curl -k -X GET --header 'Accept: application/json' -H "Authorization: Bearer ${TOKEN}"
https://$ROUTE/aiops/api/issue-resolution/mime/v1/customisation/severity -H "accept: application/json" -H "X-
TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255"
```

The following example shows a response(200) for the GET request:

```
{
"severities": [
   {
      "severity": 2,
      "weight": 20
   }
]
}
```

Use a POST request **/customisation/severity** to update(replace) the probable cause to use the array of severities. The request body is a JSON array of severities that overwrites the current array of severities.

The following example shows a response(200) for the POST request:

```
{"Result":"Success"}
```

Use a DELETE request **/customisation/severity** to delete all the severities for the specified tenant ID.

The following example shows a response(200) for the DELETE request:

```
{"Result":"Success"}
```

```
/customisation/scoring/config
```

Use a GET request **customisation/scoring/config** to get the scoring configuration parameters for the specified tenant ID. If the API request is successful(200 response), the response includes a list of the scoring configuration attributes that are turned on or off.

```
curl -k -X GET --header 'Accept: application/json' -H "Authorization: Bearer ${TOKEN}"
https://$ROUTE/aiops/api/issue-resolution/mime/v1/customisation/scoring/config -H "accept: application/json"
-H "X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255"
```

The following example shows a response(200) for the GET request:

```
{
"pathCaculationEnabled": true,
"wordEnabled": true,
"severityEnabled": true,
"causeScoreEnabled": true,
"itnmproccessing": true,
"itnmCausesWeight": 110,
"firstBoost" : true,
"firstBoostWeight" : 100,
}
```

For more information about probable cause ranking and probable cause determination, see Probable cause ranking

Next steps

For more information about probable cause, see Probable cause in ChatOps.

For more information about the available APIs and related Swagger documentation, see the additional topics under APIs. For example, for more on access tokens, see Accessing APIs.

Change Risk REST API

This documentation is for the IBM Cloud Pak for AIOps Change Risk REST API.

Endpoint URLs

Change Risk REST API root URL:

```
https://cpd-{CLUSTER_NAMESPACE}.apps.{CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.
{CLUSTER_TERMINAL_DOMAIN}/aiops/api/ai-manager/change-risk/v1
```

/change_requests/risk_assessment

POST

Get the risk assessment by providing a change request ticket information.

Parameters

- include_explanation : true or false for turning on/off explanation. Explanation is included by default.

Responses

Table. Post /change_requests/risk_assessment responses

Code	Description
200	Response
400	Bad request
404	Not found
500	Internal server error
504	Internal processing timeout
default	unexpected error

Example

The sample curl command resembles the following code:

```
CLUSTER_URL=https://cpd-cp4waiops.apps.mycluster.mydomain.myco.com/aiops/api/ai-manager/change-risk/v1
curl -X POST $CLUSTER_URL/change_requests/risk_assessment?include_explanation=true \
-H "Authorization: Bearer $ACCESS_TOKEN" \
-H 'Content-Type: application/json' \
--data '{
   "number":"CHG0775062",
   "sys_domain":"crn:v1:bluemix:public::mil01::::",
   "business_service":"Network",
   "short_description":"is DATACENTER 2020-05-17 08:29:00 UTC",
   "description":"description text field",
   "justification":"none",
   "backout_plan":".",
   "impact":"No",
   "reason":"Replace server due to hardware issue.",
   "state":"Closed",
   "type":"softlayer",
   "sys_created_by":"SoftLayer Jira.SA",
   "assigned_to":"name@email.com",
   "close_code":"successful",
   "close_notes":"Change was completed successfully"
}'
```

The data field above must validate against a change record schema. The response will look similar to the following:

```
{
   "id":"77f3f724727811eb9e81c8e0eb1474cb",
   "prediction_text":"Prior ticket data insights classify this as a high risk change.",
   "prediction_confidence":0.979616,
   "explanation":{
      "words":{
         "test":0.1,
         "run":0.073,
         "pod":0.059,
         "node":0.056,
         "error":0.041,
         "release":0.036,
         "remove":0.027,
         "backout":0.025,
         "fail":0.023,
         "plan":0.017,
         "utc":0.017,
         "system":0.013,
         "previous":0.012
      },
      "relevant_change_tickets":[
         {
            "number":"CHG0675192",
            "sys_domain":"crn:v1:bluemix:public::mil01::::",
            "business_service":"Network",
            "short_description":"is DATACENTER 2020-05-17 08:29:00 UTC",
            "description":"description text field",
            "justification":"none",
            "backout_plan":".",
            "impact":"No",
            "reason":"Replace server due to hardware issue.",
            "state":"Closed",
            "type":"softlayer",
            "sys_created_by":"SoftLayer Jira.SA",
            "assigned_to":"name@email.com",
            "close_code":"successful",
            "close_notes":"Change was completed successfully"
            "induced_incidents":[
               {
                  "description":"description text",
                  "number":"INC2468135",
                  "severity":"1 - High"
               }
            ]
         }
      ],
      "ticket_relevance_level":{
         "CHG0675192":10.0
      },
      "ticket_similarity_level":{
         "CHG0675192":1.0
      }
```

```
    }
}
```

/feedback/types/risk_assessment

POST

Provide feedback for the corresponding change risk assessment

Responses

Table. Post /feedback/types/risk_assessment responses

Code	Description
200	Response
400	Bad request
404	Not found
500	Internal server error
default	unexpected error

/feedback/types/risk_assessment_explanation

POST

Provide feedback for the corresponding change risk assessment

Responses

Table. /feedback/types/risk_assessment_explanation responses

Code	Description
200	Risk assessment response
400	Bad request
404	Not found
500	Internal server error
default	unexpected error

/healthcheck

GET

Gets the health of the server.

Responses

Table. Get healthcheck responses

Code	Description
200	OK
500	Internal server error

Models

Change

Table. Change models

Name	Type	Description	Required
number	string		Yes
sys_domain	string		No

Name	Type	Description	Required
business_service	string		No
short_description	string		No
description	string		No
justification	string		No
backout_plan	string		No
impact	string		No
reason	string		No
state	string		No
type	string		No
sys_created_by	string		No
assigned_to	string		Yes
close_code	string		No
close_notes	string		No

Incident

Table. Incident models

Name	Type	Description	Required
number	string		Yes
description	string		Yes
severity	string		No

Feedback

Table. Feedback model

Name	Type	Description	Required
user_response	string		Yes

FeedbackRiskAssessment

Table. FeedbackRiskAssessment models

Name	Type	Description	Required
assessment_id	string		Yes
feedback	object		Yes

FeedbackRiskAssessmentExplanation

Table. FeedbackRiskAssessmentExplanation models

Name	Type	Description	Required
assessment_id	string		Yes
relevant_ticket_number	string		Yes
feedback	object		Yes

RiskAssessment

Table. RiskAssessment models

Name	Type	Description	Required
number	string		No
id	string		Yes
prediction_text	string		Yes
prediction_confidence	float		Yes
change_risk_score	float		Yes
explanation	object		Yes

Explanation

Table. Explanation models

Name	Type	Description	Required
words	object		Yes
relevant_tickets	[object & Change]		No
ticket_relevance_level	object		No

Name	Type	Description	Required
ticket_similarity_level	object		No

RelevantTickets

Table. RelevantTickets model

Name	Type	Description	Required
RelevantTickets	array		

ErrorContainer

Table. ErrorContainer models

Name	Type	Description	Required
errors	[object]		No
trace	string	*Example:* `"9daee671-916a-4678-850b-10b911f0236d"`	No

Errors

Table. Errors model

Name	Type	Description	Required
Errors	array		

Error

Table. Error models

Name	Type	Description	Required
http_code	integer	*Example:* `500`	No
code	string	*Example:* `"missing_field"`	No
message	string	*Example:* `"The` id `field is required."`	No
more_info	string		No

Metric REST API

The Metric API allows for the posting of metric data for anomaly detection and forecasting.

Endpoint URLs

Metric Ingestion API

```
cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/app/metric-api/v1/metrics
```

Metric Retrieval API

```
cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/app/metric-api/v1/metrics
```

Metric Forecasting API

```
cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/app/metric-api/v1/forecast
```

Authentication

The API uses Platform UI (Zen) authentication, so you either need a JWT token or ZenApiKey to be authenticated. The token or key is used to determine whether a user or service ID has access when you use the API.

Error handling

Metric Ingestion API error handling

Table. Metric ingestion API error codes

HTTP Error Code	Description	Recovery
200	Success	Success.
400	Bad Request	Invalid value in JSON.
403	Forbidden	The supplied authentication is not authorized to access '{namespace}'.
500	Internal Server Error	An error occurred processing your request.

Metric Retrieval API error handling

Table. Metric retrieval API error codes

HTTP Error Code	Description	Recovery
200	Success	Success.
400	Bad Request	Invalid parameters.
403	Forbidden	The supplied authentication is not authorized to access '{namespace}'.
500	Internal Server Error	An error occurred processing your request.

Metric Forecasting API error handling

Table. Metric forecasting API error codes

HTTP Error Code	Description	Recovery
200	Success	Success.
204	No Content	Success, no data is found for provided attributes.
400	Bad Request	Invalid value in JSON.
401	Unauthorized	You are not authorized to make this request.
403	Forbidden	The supplied authentication is not authorized to access '{namespace}'.
500	Internal Server Error	An error occurred processing your request.

Ingest data

To inject data for metric analysis, you need to first log in to your cluster and authenticate for using APIs:

1. Log in to your cluster and authenticate your admin user credentials for running API commands:
    1. Get the route for loading your metric data by running the following command:

        ```
        oc get route | grep ibm-nginx-svc | awk '{print $2}'
        ```

    2. Get your admin username by running the following command

        ```
        oc -n ibm-common-services get secret platform-auth-idp-credentials -o
        jsonpath='{.data.admin_username}' | base64 -d; echo
        ```

    3. Get the password for your admin username by running the following command:

        ```
        oc -n ibm-common-services get secret platform-auth-idp-credentials -o
        jsonpath='{.data.admin_password}' | base64 -d; echo
        ```

    Note: Two methods of authentication are available:

    - JWT Bearer Token

      This token is temporary and lasts about 4 hours.

    - Platform UI (ZenApiKey) API key

      This key remains until deleted.

    For more information about API authentication, see Accessing APIs

2. Encode your admin username and API key with Base64 encoding:

    ```
    ZENAPIKEY=$(echo "<username>:<API_key>" | base64 -w 0); echo $ZENAPIKEY
    ```

3. Get and export the route to use to send in your training data:

    ```
    export ROUTE=$(oc get route | grep ibm-nginx-svc | awk '{print $2}')
    ```

4. Run a cURL to load the training data file by using the route and Platform UI API Key (ZenApiKey)

    ```
    curl -k -v -X POST "https://${ROUTE}/aiops/api/app/metric-api/v1/metrics" --header 'Content-Type:
    application/json' --header "Authorization: ZenApiKey ${ZENAPIKEY}" --header 'X-TenantID: cfd95b7e-3bc7-
    4006-a4a8-a73a79c71255' --data @<FILEPATH/FILE>.json
    ```

Where **<FILEPATH/FILE>** is the full path to the JSON file that includes your training data to load.

To ensure that data is injected, run a cURL command to return data with the API:

```
curl -v "https://${ROUTE}/aiops/api/app/metric-api/v1/metrics?resource=MyApp&group=Latency&metric=calls.sum"
--header "Authorization: ZenApiKey ${ZENAPIKEY}" --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
--insecure
```

JSON schema for metrics data

JSON Payload:

```
{
    "groups": [
        {
            "timestamp": <timestamp>,
            "resourceID": "<resource ID>",
            "metrics": {
                "<kpi-1>": <value-1>,
                "<kpi-2>": <value-2>,
                ...
                "<kpi-n>": <value-n>
            },
            "attributes": {
                "group": "<metric namespace>",
                "node": "<top-level resource>"
            }
        }
    ]
}
```

- **timestamp** - The epoch time in milliseconds (13 digits) or ISO-8601.
- **resourceID** - The resource that is being measured.
- **metrics** - A map of metric name to metric value.
- **attributes** - Anything that helps provide context to the user.
- **group** - Field that you can use to namespace a metric. For example, "customer1". Therefore, group helps distinguish metrics names.
- **node** - The top-level resource.

Example JSON Payload:

```
{
    "groups": [
        {
            "timestamp": "1642516320000",
            "resourceID": "MyApp",
            "metrics": {
              "calls.sum": 476.0,
              "latency.mean": 1.6722689
            },
            "attributes": {
              "group": "Latency",
              "node": "MyApp"
            }
        }
    ]
}
```

Note: The **attributes.group**, **resourceID**, and the **metrics** (for example, **metrics.access-denied**) fields, along with the **tenantID** (part of the header in the HTTP request) make up the unique identifier for a KPI. Therefore, as part of the metric anomaly processing and detection, any data points with different values for these fields are considered as different metrics. For example, consider the following two data points:

```
{
  "groups": [
    {
      "timestamp": 1634846130000,
      "resourceID": "10.10.10.2",
      "metrics": {
        "access-denied": 14977
      },
      "attributes": {
        "group": "Org1",
        "node": "firewall-1"
      }
    },
    {
      "timestamp": 1634846130000,
      "resourceID": "10.10.10.2",
      "metrics": {
```

```
        "access-denied": 379
      },
      "attributes": {
        "group": "Org2",
        "node": "firewall-1"
      }
    }
  ]
}
```

These two data points have the same **timestamp** value (**1634846130000**) and **resourceID** value (**10.10.10.2**), but have two different group attribute names (**Org1** and **Org2**). This difference causes the data points to be treated as different metrics.

The JSON metric field for correlating a metric anomaly with an element in a resource group depends on the data source for the corresponding topology data. For example, for topology data that is ingested through an Instana connection, the following matching rule is used for correlation: **${name}:::${uniqueId}**. As another example, for topology data that is ingested through a file observer job, the **resourceID** field is used as the identifier for topology correlation.

Unpegging, accumulator and counter attributes: For metrics that count upwards only from when the system starts, the analytics wants to work on the delta values rather than the raw values. To identify which metrics are pegs, accumulators, or counters in metric anomaly detection, set an attribute per metric group, for example: **"accumulators":"TotalConnections,SuccessfulLoginCount"**

Aggregation attributes: Average is the default aggregation type, except for accumulators, where sum is the default. To identify which metrics are to be aggregated with sum in metric anomaly detection, set an attribute per metric group, for example: **"sumAggregator":"newConnections,Logins"**

Runbook automation REST API

Runbook Automation is responsible for the management of runbooks and runbook instances. Runbooks are structured operational procedures which can be created, changed, and executed. They might include several types of automation and run manual, semi-automated, or fully-automated. Runbook instances are the execution records of Runbooks. They contain the status and details of one specific execution and can be changed to step through an execution and finish it. Additionally other administrative functions are presented in the API, including configuration management. To send API calls, create an API key in the application and find detailed API documentation inside the Swagger document.

Endpoint URLs

Runbook API

It is used to manage runbooks.

cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/story-manager/rba/v1/runbooks

Runbook instance API

It is used to manage runbook executions.

cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/story-manager/rba/v1/runbookinstances

Configuration API

It is used to manage Runbook Automation configuration.

cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/story-manager/rba/v1/configuration

Tag API

It is used to retrieve runbook tags.

cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/story-manager/rba/v1/tags

Usage Statistics API

It is used to retrieve Runbook Automation usage statistics.

cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/story-manager/rba/v1/statistics/runbookinstances

Status API

It is used to retrieve the status of the Runbook Automation deployment.

```
cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/story-manager/rba/v1/status
```

User API

It is used to retrieve a list of all users who have a given role.

```
cpd-aiops.apps.{OC_CLUSTER_NAME}.{CLUSTER_DOMAIN_PREFIX}.{CLUSTER_TERMINAL_DOMAIN}/aiops/api/story-manager/rba/v1/users
```

Authentication

The API uses Platform UI (zen) authentication which requires a JWT token to be authenticated. The token is used to determine whether a user or service ID has access when they use the API.

Error handling

The Runbook Automation API returns one of the following HTTP responses for every API call.

Table. Runbook automation API error codes

HTTP Error Code	Description	Recovery
`200`	Success	The call was successful. See the output for the requested data.
`204`	No Content	Action completed successfully, no data is returned by the server.
`400`	Bad Request	The input was invalid. Read the error message and adjust your input before trying again.
`401`	Unauthorized	The request was not authorized. Make sure your API key is valid and the user and password are entered correctly.
`403`	Forbidden	The supplied authentication is not authorized to access the given API.
`500`	Internal Server Error	An internal error happened. A correlationId is provided. Include this ID in any request towards IBM support.

Application and topology APIs

IBM Cloud Pak® for AIOps offers a set of tools and APIs that you can use to predict, communicate, and resolve events before they become serious problems for your organization.

To learn more about how topologies are displayed in the UI, review the topology viewer documentation. For more information, see Topology viewer.

API authentication

For more information on how to authenticate to use APIs, see Accessing APIs.

Access to URL routes

For more information on how to access the URL routes for the topology Swagger documentation, see Enabling access to URL routes.

Access the topology service Swagger through port forwarding

- Accessing topology Swagger

Swagger documentation

Review the following Swagger documentation to use Topology APIs for configure File Observer and Rest Observer jobs.

- Topology File Observer API
- Topology Rest Observer API

To learn more about how to use these jobs, such as for including custom data in topologies, see:

- Configuring REST Observer jobs
- Configuring File Observer jobs

Troubleshooting: Swagger UI not responding during high-volume query

When querying for resources using **`GET /resources`**, the Swagger UI may become unresponsive if there are in excess of a thousand results.

- **Cause:** This is a Swagger limitation.
- **Workaround:** None. Avoid high-volume Swagger queries.

API reference documentation

To learn more about using the topology APIs, including terminology and properties, see Topology API reference.

Accessing Topology service Swagger UI

You can access the Topology service Swagger using port forwarding.

Before you begin

You need to be an administrator with **`oc client`** access in order to **`get`** secrets, services, and to enable port-forwarding in the ASM namespace.

About this task

A simple way to access the Topology service Swagger is to use the **`oc port-forward`** command. You can then access Swagger on your local browser through **`localhost`**. **Note:** Port forwarding will be enabled temporarily only. When there is no traffic, it will automatically be closed.

- To log on, use **`oc login`**
- To obtain the service name, use **`oc get service`**
- To enable port forwarding, use **`oc port-forward service/aiops-topology-topology 8080`**

Enable port forwarding

See the following steps as an example.

1. Log on using the **`oc login`** command.
2. Find the secret storing the credentials for Swagger using the **`oc get secret | grep topology-asm-credentials`** command.

   System output:

   ```
   aiops-topology-asm-credentials                                    Opaque
   2      3h54m
   ```

3. Find the topology service secret using the **`oc get service | grep topology-topology`** command.

   System output:

   ```
   aiops-topology-topology                                  ClusterIP   1.2.3.4    <none>
   8080/TCP,8081/TCP                                                    3h49m
   ```

4. Enable port forwarding using the **`oc port-forward service/aiops-topology-topology 8080`** command.

   System output:

   ```
   Forwarding from 127.0.0.1:8080 -> 8080
   Forwarding from [::1]:8080 -> 8080
   ```

   Note: If the port is not accessed for a while, the connection is closed:

   ```
   Forwarding from [::1]:8080 -> 8080
   E0622 07:57:52.101053   25681 portforward.go:233] lost connection to pod
   ```

5. Now access the topology service swagger api in a browser on the same host as the one the oc commands were run on, at the following URL **`https://localhost:8080/1.0/topology/swagger`**
6. Authorise using the credentials from the secret obtained in step 3.

Enabling access to URL routes

To access the URL routes for the topology Swagger documentation, you update the IBM® Netcool® Operations Insight® Operator configuration.

About this task

The URL to access an observer API consists of **{HOST}/{PATH}** and the Swagger UI URL consists of **{HOST}/{PATH}/swagger**.

Enable access to the URL routes

1. Add the following code to the spec section of the custom resource of the operator to enable the routes.

   The following code will expose all of the observers for both **full** and **listen** jobs.

   ```
   spec:
     helmValuesASM:
       global.enableAllRoutes: true
   ```

2. Use the *oc get route | grep* command to obtain an observer API URL:

   The following example uses the AppDynamics Observer:

   ```
   oc get route | grep appdynamics
   ```

   Output:

   ```
   NAME                                        HOST/PORT
   PATH                                SERVICES                                        PORT    TERMINATION
   WILDCARD
   aiops-topology-appdynamics-observer       aiops-topology-appdynamics-observer-aiops.apps.myibm.com
   /1.0/appdynamics-observer       aiops-topology-appdynamics-observer       9148    reencrypt/Redirect
   None
   ```

Examples

Example cURL command to the AppDynamic Observer (or job) endpoint

```
curl -k -X GET --header 'Accept: application/json' -H "Authorization: Bearer ${TOKEN}" https://aiops-
topology-appdynamics-observer-aiops.apps.myibm.com/1.0/appdynamics-observer/jobs --header 'X-TenantID:
cfd95b7e-3bc7-4006-a4a8-a73a79c71255'
{"_items":[],"_limit":100}
```

Note: To obtain your access token, see Accessing APIs.

Example Swagger UI URL: `https://aiops-topology-appdynamics-observer-aiops.apps.myibm.com/1.0/appdynamics-observer/swagger`

Next steps

For more information on the available API and related Swagger documentation, see the additional topics under APIs.

Topology API reference

The following topics explain the common terminology, properties, types, and more for the available topology APIs.

Terminology

In addition to the following terms, ensure that you are familiar with the important terms that are defined within the Glossary.

resource: A resource is a node in an interconnected topology, sometimes also referred to as a vertex, or simply a node. It can be anything in a user-specific topology that is designated as such, for example a hardware or virtual device, a location, a user, or an application.

edge: An edge is a relationship between resources, also simply referred to as the 'link' between resources. Edges have a label, which allocates them to a family of edges with specific behavior. It also governs how they are displayed in the UI, and an edgeType, which defines the relationship in real terms.

tenant: A tenant is represented by a globally unique identifier, its tenant ID. The default tenant ID is: cfd95b7e-3bc7-4006-a4a8-a73a79c71255

provider: A provider is usually a single data source within the scope of a tenant. The provider is a namespace for the resource's uniqueId. That uniqueId property is unique only within the scope of a provider.

status: Status is a property of one or more resources, and a single resource can have different types of status. Each status can be in one of three states: 'open', 'clear' or 'closed'. The severity can be one of the following ratings: 'clear', 'information', 'indeterminate', 'warning', 'minor', 'major' or 'critical'.

The status of a resource is derived from events, or if the resource is retrieved through the status service. It can also be supplied when resources are posted to the topology service.

Reference documentation

Properties

The Topology Service has two categories of properties, generic and user. Generic properties have fixed data types, while user-defined properties do not.

Edge labels

The topology service defines a family of labels for the edges it supports.

Edge types

All edges created in the Topology Service should define an edge **type**. The following section lists the edge types that are associated with each of the public-facing edge labels. If none of the default edge types suffice, you can create custom edge types.

Entity types

Use the Topology Service to group together resources of the same type using the **`entityTypes`** property, and identify them in the UI by their icon. Usually a resource has only a single entity type but as the property is a *Set*, it is possible for a resource to have more than one entity type. A number of pre-defined entity types are supplied. In addition, you can create more entity types as required.

Status (and state)

Resources have specific statuses, and in turn each status has a state of open, clear or closed.

Timestamps

Both vertex and edge graph elements can have multiple timestamps, as are documented here.

Topology examples

You can review examples that describe best-practice guidelines to provide you with practical information, such as implementation examples and code samples, that you can use to get started.

Properties

The topology API has two categories of properties, generic and user. Generic properties have fixed data types, while user-defined properties do not.

- Some properties have special semantics, as described in **Generic properties**.
- The pre-defined properties are listed in the **Pre-defined properties** table.
- You can add any property as required, as described in **User properties**.

Generic properties

Generic properties are few and constrained to a fixed data type. They can also be subdivided into those which are read/write and those which are read-only.

- **`uniqueId`**

  The uniqueId is the string that is used to match resources from the same provider. It could be, for example, a UUID by which the provider can look up its own local data store for information about that device.

  If you send the same resource with the same uniqueId and the same provider more than once, it will be treated as the same resource.

  The provider is a project (namespace) for the resource's uniqueId. That uniqueId property is unique only within the scope of a provider.

- **`matchTokens`**

  These tokens are used to store strings, which are significant to that resource, and might match it to events.

- **`name`**

  This string does not have to be unique, and it should be short and memorable.

- **`tags`**

  Tags can be used to search for resources in the topology view, and to group resources. For more information on configuring tag rules and tag-based templates, see the following sections:

  - Defining rules
  - Group templates

- **`entityTypes`**

  These are defined as a set, though with usually only a single member, of one or more types this resource represents.

  For more information on entityTypes, see the Entity types topic.

Predefined properties

Table. Predefined properties

Name	Type	Cardinality	Alias	Read-only	Indexed
age	integer	single		no	y
aliasIds	ID	set	_aliasIds	yes	n
beginTime	long	single	_startedAt	yes	y
changeTime	long	single	_modifiedAt	yes	n
createTime	long	single	_createdAt	yes	y
deleteTime	long	single		yes	n
description	string	single		no	y
edgeTenantId	ID	single	_edgeTenantId	yes	n
edgeType	string	single	_edgeType	yes	n
endTime	long	single	_deletedAt	yes	y
entityTypes	string	set		no	y
eventId	string	single		yes	n
eventManager	string	single		yes	n
expireTime	long	single	_expiredAt	yes	n
geolocation	GeoLocation	single		no	n
hasState	string	single		yes	n
icon	string	single		no	n
id	long	single		yes	n
keyIndexName	string	single		yes	n
label	string	single		yes	n
matchTokens	string	set		no	y
name	string	single		no	y
observedTime	long	single	_observedAt	yes	n
operation	string	single		yes	n
partOfExternal	Boolean	single		yes	n
prevBeginTime	long	single		yes	y
providerId	ID	single		yes	n
providerName	string	single		yes	n
reconciliationTokens	string	set		yes	n
referenceId	ID	single		yes	y
referenceNo	long	single		yes	n
serializedHashMap	HashMap	single		yes	n
severity	string	single		no	n
speed	long	single		no	y
statusType	string	single		yes	n
tags	string	set		no	y
tenantIds	ID	set	_tenantIds	yes	y
uniqueId	string	single		no	y

Name	Type	Cardinality	Alias	Read-only	Indexed
uuid	ID	single	_id	yes	y
version	string	single		no	y
vertexType	string	single		yes	y

User properties

User-defined properties are free-form, and are not constrained by any data type. You can add new user properties as needed.

You can define any custom properties, such as, **`ipAddress`**.

Note: All user-defined properties such as **`ipAddress`**s are not in the generic set, and are stored as a serialized **`blob`** data type instead. The implication of this serialized data type storage convention is that these properties cannot be filtered, as they are incompatible with the **`_filter`** query parameter used in the REST API.

JSON record depicting both types of properties

The following picture depicts how the properties in a JSON record from an Observer can be split into both generic and user properties.

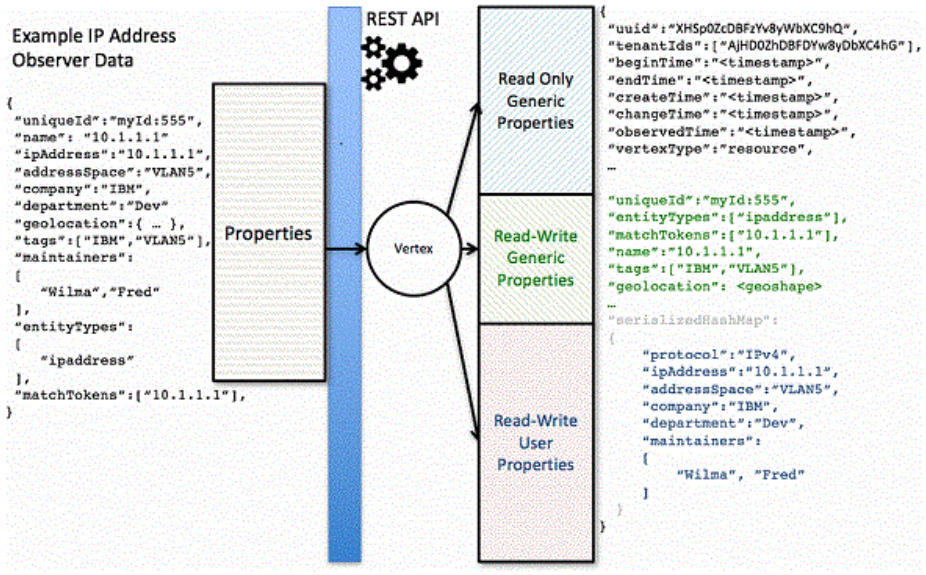


Figure. JSON record properties

Edge labels

The topology service defines a family of labels for the edges it supports.

Note: Most interactions with edges in the topology are with edge *types* rather than edge *labels*. These types can be conceptualized as instances of the edge label classes, and are documented separately. For more information, see Edge types.

- **`aggregation`**

  A relationship where the target vertex can exist independently of the source, but is needed for the source to function.

  A 'parent-child' relationship where the parent (source) is aggregating the children (target).

  The type of aggregation is determined by the value of the edge type.

  Use this edge label to represent a resource in the UI that is composed of various elements, for example a book **`contains`** words.

  - The direction is always from parent to child.
  - Children can have multiple parents.

  For more information on the types that associated with this label, see Edge types.

- **`association`**

A ‘weak’ relationship where both source and target vertex can exist independently and each does not require the other to function.

The specific type of association is determined by the edge type.

Use this edge label to represent a general relationship between vertices in the UI, for example a person **`has`** a house.

The label itself has no direction, but a direction could be implied by the edge **type** used.

For more information on the types that associated with this label, see Edge types.

- **`composition`**

  A ‘part-whole’ relationship where the child (source) is a part of the parent (target).

  Use when you need to create a short-cut from a node to its root, bypassing fine-grained relationships, for example a component is **partOf** a device.

  - The direction is always child to parent.
  - Children cannot have multiple parents.
  - Composition relationships are hidden from the element-to-element view. They are used to construct the element-to-host view.
  - **partOf** is the only edge type you can use for the 'composition' label, and is reserved for its exclusive use.
- **`dataFlow`**

  A dataFlow label represents a data flow between a pair of vertices.

  The specific type of data flow is qualified by properties on the edge type.

  Use this label when you need to represent any form of data flow in the UI, for example a person emailing another person.

  The label itself has no direction, but a direction can be implied from the edge type used.

  For more information on the types that associated with this label, see Edge types.

- **`dependency`**

  A 'strong' relationship where the source depends on the target and cannot operate independently of it, and is not integral to it.

  The specific type of dependency is determined by the edgeType.

  Use this label when you need to represent the dependency of one resource on another in the UI, for example an application dependsOn a database.

  The direction is always from the dependent resource to the independent resource.

  For more information on the types that associated with this label, see Edge types.

Edge types

All edges created in the topology should have an edge **type** defined. Review the following lists of the types that are associated with each edge label. If none of the default types suffice, you can create your own custom types.

Edge **types** can be thought of as being instances of the edge **label** classes. Most interactions with edges in the topology are with *types* rather than *labels*.

Edge types for labels

Types for the aggregation label

Edge type	Description	Example
contains	The source resource is considered to contain the target resource	Slot **`contains`** a card
federates	The source resource is a federation of the target resources	Database **`federates`** nodes
members	The source resource has the target resources as its members	Subnet **`members`** are IP addresses

Types for the association label

Edge type	Description	Example
aliasOf	Depicts that one resource is an alias of another; potentially from different providers	FQDN1 is an **`alias`** of FQDN2
assignedTo	When one resource is assigned to another	The alarm is **`assingedTo`** an operator
attachedTo	When one resource is attached to another	The plug is **`attachedTo`** to the cable

Edge type	Description	Example
classifies	When one resource is used to classify another	The government **classifies** the document
configures	When one resource configures or provides configuration for another resource	The file **configures** the application
deployedTo	Describes when one resource is deployed to another resource	The application was **deployedTo** the server
exposes	When one resource exposes another	The application **exposes** the interface
has	Generalized relationship when one resource possesses, owns, or has another resource	Host **has** component
implements	When one resource implements another	The class **implements** the interface
locatedAt	When one resource is physically located in or at another resource's location	Host is **locatedAt** data center
manages	When one resource manages another	The boss **manages** the employee
monitors	When one resource monitors another	The application **monitors** the host
movedTo	When one resource moves to a new and different resource	The service has **movedTo** the host
origin	Indicates the origin of a Vertex	Device's **origin** is a vendor
owns	Indicates ownership of one resource by another resource	The user **owns** the server
rates	Can be used when one resource rates another	The manager **rates** the employee
resolvesTo	When one resource resolves to another	The hostname **resolvesTo** an address
realizes	When one resource realizes another	The hypervisor **realizes** the virtual machine
segregates	When one resource segregates another	The firewall **segregates** the network
uses	When one resource takes, employs or deploys another resource as a means of achieving something	Application **uses** this disk

Types for the dataFlow label

Edge type	Description	Example
accessedVia	One resource is accessed through another, typically remote.	Server is **accessedVia** a FQDN
bindsTo	A network layering relationship such that the source 'runs on top' of the target.	Logical interface **bindsTo** a physical port
communicatesWith	A relationship whereby one resource communicates with another	The sensor **communicatesWith** an application
connectedTo	A relationship whereby one resource is connected to another	The laptop is **connectedTo** the switch
downlinkTo	One resource is down-linked to another	The controller has a **downlinkTo** the sensor
reachableVia	A resource is reachable through another resource	The network is **reachableVia** the gateway
receives	A resource receives data from another	The Mail server **receives** an email
routes	A relationship whereby one resource routes data for another	The device **routes** the data
routesVia	A relationship whereby data from one resource routes through another	The traffic **routesVia** the device
loadBalances	One resource which balances load for others	The load balancer **loadBalances** to servers
resolved	When one resource resolved something for another	DNS server **resolved** IP address
resolves	Represents that one resource uses another to resolve it	FQDN **resolves** to the address
sends	A resource sends some data to another	The application **sends** an SMS message
traverses	Describes when one resource traverses another	The message **traverses** the network
uplinkTo	One resource is up-linked to another	The sensor has an **uplinkTo** the controller

Types for the dependency label

Edge type	Description	Example
dependsOn	A generic dependency between resources	One application **dependsOn** another
runsOn	A resource runs on (and therefore depends on) another resource	The service **runsOn** the host

Custom edge types

If no default edge types suitably represents a relationship, you can define a custom edge type through the **`https://localhost:8080/1.0/topology/types/edge`** POST API.

Important: A custom edge type needs to be created **before** the observation job passes any edges with the type to the topology service.

- **`edgeType`**

Required. The edgeType name, which must be unique and **cannot** match the edgeType name of a default edgeType, unless the edgeLabel also matches the corresponding default edge type's edgeLabel parameter.

Restriction: A scenario where both edgeType and edgeLabel match the fields of a default edge type is equivalent to manually creating a default edge type, which is not necessary as default edge types are created implicitly by the topology when needed.

- **`edgeLabel`**

  Required. The edgeLabel of the custom edgeType, which must have one of the following labels:

  - dataFlow
  - dependency
  - association
  - aggregation
- **`description`**

  Optional, but recommended. This should be a meaningful description of the type of relationship this edge type represents.

Example

```
curl -k -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header 'X-TenantID: cfd95b7e-3bc7-4006-a4a8-a73a79c71255' -d '{
  "edgeType": "connectedTo",
  "edgeLabel": "dataFlow",
  "description": "Default relationship between two devices that exchange data"
}' 'https://localhost:8080/1.0/topology/types/edge'
```

Entity types

The Topology Service allows you to group resources of the same type that uses the **`entityTypes`** property, and identify them in the UI by their icon.

Usually a resource has only a single entity type, but, as the property is a *Set*, it is possible for a resource to have more than one entity type. Pre-defined associations of icons to specific entityTypes are listed in the following sections. You can create additional entity types as required.

- An entityType is defined by defining a resource with that type in the entityTypes list field.
- You can associate icons with named types in the Topology Viewer.
- You can add custom icons.

Predefined entity types

An entity type is used to map the resource vertex to an icon.

Notes:

- An administrator user can create more custom resource or entity types as required. Any observer may add new types.
- The topology service API only returns the resource or entity types in use.

To assign an entityType

1. Select a resource with the provided entityType in the Topology Viewer.
2. Select the topology tools (the **cog** icon), then navigate to the resource types.
3. Find the type to be edited, and select the icon to assign it. You can find an icon either by name from the dropdown list, or by using the **View all icons** option (the **eye** icon).

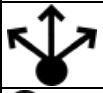
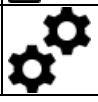
Entity type icons

The following table lists these entity types with links to their icons, if defined:

Table. Entity type icons

Entity type	Icon
aircond	
alarm	
application	
backplane	

Entity type	Icon
bridge	
building	
bypass	
card	
cell	
channel	
chassis	
circuit	
cluster	
command	
component	
computer	
container	
cpu	
credential	
daemonset	
database	
datacenter	
deployment	
device	
directory	
disk	
domain	
emailaddress	
environment	
event	
fan	
file	
firewall	
flavor	
forum	
fqdn	
gauge	
group	
host	
hsrp	
hub	
image	
input	

Entity type	Icon
ipaddress	
ipam	
loadbalancer	
location	
memory	
message	
namespace	
network	
networkaddress	
networkgear	
networkinterface	
operatingsystem	
organization	
outlet	
output	
patch	
path	
pdu	
peripherals	
person	
pod	
printer	
process	
product	
project	
psu	
resource	
router	
routetable	
routetarget	
rsm	
rule	
sector	
securitygroup	
sensor	
server	
service	

Entity type	Icon
serviceaccesspoint	
slot	
snmpsystem	
stack	
statefulset	
status	
storage	
subnet	
subscription	
switch	
swarm	
task	
tcpudpport	
template	
tenant	
tier	
tool	
ups	
variable	
version	
vlan	
vm	
volume	
vpn	
vrf	
workload	

Status (and state)

Resources have specific statuses, and in turn each status has a state of open, clear, or closed.

Status

A single status can affect multiple resources, and a single resource can have multiple different statuses. For example, a single *link down* event can generate the status for both the interface resource and the host resource, or, a single host might have *CPU* or *disk usage* status in addition to any *link down* status.

Resource status can be viewed in the Topology Viewer. An 'open' event is one with a severity other than 'clear'.

Important: When modeling resources, you must consider Status assignment from events.

Tip: The Topology Service stores the event **Severity**, and nodes in the UI are colored based on severity, which always has one of the following definitions:

- clear
- indeterminate
- information

- warning
- minor
- major
- critical

Look at the severity icons in the topology viewer reference topic Severity icons table.

Status assignment

The status of a single resource can be supplied, alongside other resource properties, when creating or updating a resource. Alternatively, an event can generate the status of one or more resources.

Remember: A status **always** has one of the following three states:

- **`Open`**

  Always has a severity other than 'clear' to indicate an active issue that can require your attention.

- **`Clear`**

  Working as expected. Also has a severity of 'clear'.

- **`Closed`**

  No longer active or relevant, a deleted event. Also has a severity of 'clear'.

Status assignment from events

The status service receives events and tries to find matching resources in the topology service. A resource with no match tokens defined will not have events matched to it, but if found, the status of those resources is set from the event data. The default fields of the alert used to look up resource matchTokens are *resource.sourceId* and *resource.name*. The assigned status depends on the following event data:

- **`matchTokens`**

  This property must be used to list any data that can identify (match) the resource.

  Each field may be globally unique, or may be unique within the scope of a composition. In other words, a resource modeled as a composition relationship, such as **`partOf`**, can be distinguished from other children within the composition using these fields.

  For example, either a globally unique and fully qualified domain name or IP address, or a locally unique interface name (that is, local within the scope of the host), can be used to identify the resource.

- **`partOf`** composition relationship

  The status service uses composition relationships to match fields that are unique only within the scope of a parent.

  For example, an IP address can be used to find a main node, and an interface name can be used to identify an interface within that main node.

- **Note:** If two different resources have the same **`matchTokens`** value, an event cannot be assigned to them unless they have been merged to become members of the same composite, **or** they have a parent-child **`partOf`** relationship.

Access scope

The scoping functions allows for preferential matching of AIOps alerts. If similar resources and alerts in your environment exist in multiple different scopes, you can tag them with scope-specific information to allow a unique match to be made.

Such a scenario can occur (for example) under the following conditions:

- The same applications are running in test, staging and production environments.
- The same deployments are running in multiple namespaces in a single Kubernetes cluster.
- The same components are running in cloned Openstack projects.
- The same private IPs exist in multiple routing domains.

If an alert matches duplicate resources, a (case-insensitive) match of the alert scope against the resource scope is attempted.

Resource scope: Resources have an *accessScopeTokens* list property containing the following information:

- The provider of the resource, as defined by the observer job
- Optional additional scope supplied in the observer job's *Access Scope* CSV string parameter
- Scope explicitly added by some observers, such as the Kubernetes observer's *data_center* and *namespace* job parameters

Alert scope: The alert fields to use for scope are configurable.

- You can use the **AIOPS_SCOPING_FIELDS** variable defaults (*accessScope*, scopeId, *location*, *cluster*) to deuplicate resources when there are multiple matches on matchTokens.
- The values of these fields are compared with the resource property *accessScopeTokens* , with preferential matching based on case-insensitive comparison.

Troubleshooting alert lookup failures

1. Check for a unique resource match at `GET /resources?_filter=matchTokens:{alertFieldToLookup}`
2. If the alert is older than the resource, it might not have existed when the alert processing looked it up.

Timestamps

Both vertex and edge graph elements can have multiple timestamps.

The following formats apply across all Topology APIs when specifying timestamps:

- Millisecond Epoch

  Example:

  ```
  date "+%s%3N"
  1655991641000
  ```

- RFC-3339 / ISO-8601

  Format:

  ```
  YYYY-MM-DDThh:mm:ss.sssZ
  ```

  Example:

  ```
  date --universal "+%FT%T.%3NZ"
  2022-06-23T13:40:41.000Z
  ```

- **`beginTime`** - The beginTime timestamp records the beginning of a period of time for which the resource was valid, with endTime marking the end of that period.

  Tip: A resource can have multiple begin times in its historic record, and gaps can happen in that record if the resource was offline for periods.

  - All resources and historic resources that are representations of the same thing have a distinct beginTime
  - Resource beginTime together with endTime is used in historic graph traversals, that is, when the _at parameter is supplied. The period during which a resource is valid is defined as:

    ```
    atTime >= beginTime && atTime < endTime
    ```

  - A vertex or edge, which has the beginTime equal to the endTime, can be used to store audit information, such as the provider which deleted a resource. However, because it takes up zero time it does not form part of the history and is ignored by the preceding equation.

- **`prevBeginTime`** - If history exists for a resource, then this property is set to the beginTime of the most recent historical resource.
- **`changeTime`** - The changeTime timestamp records when the properties of the element last changed. The value can be less than the observedTime, which is updated on a POST or PUT even if no property values have changed.
- **`createTime`** - The createTime timestamp records when the element was first created in the topology service.
  - Historical resources do not store createTime, as it is shared with the anchor.
  - createTime is needed when looking for something older than 30 days – that is, when there is no beginTime this old because the historical resources have timed out.
- **`endTime`** - The endTime timestamp records when the element was deleted.
  - All resources and historic resources that are representations of the same thing have a distinct endTime.
  - Resource endTime is used in historic graph traversals, that is, when the _at parameter is supplied.
  - For current resources, endTime is LONG _MAX. This is sometimes hidden through the REST API.
- **`observedTime`** - The observedTime timestamp records when the element was last observed, that is, when data was last input to the topology service for the element.

Topology examples

The following examples describe best-practice guidelines to provide you with practical information, such as implementation examples and code samples, that you can use to get started.

Restriction: The examples provided here must be amended and adjusted to suit your own implementation.

Virtual machine example

One of the most important goals of IBM Cloud Pak® for AIOps is to support the assurance and provisioning of modern IT, network, and storage environments. These environments all make extensive use of increasingly nested virtualization technologies that need to be modeled. The following example introduces such an IT Virtualization scenario, and describes an OpenStack response that provides a solution.

Physical device example

The following example of an IBM Tivoli Network Manager environment accessed through IBM Cloud Pak for AIOps provides insights into any environment that uses physical network devices.

Virtual machine example

One of the most important goals of IBM Cloud Pak® for AIOps is to support the assurance and provisioning of modern IT, network, and storage environments. These environments all make extensive use of increasingly nested virtualization technologies that need to be modeled. The following recipe introduces such an IT Virtualization scenario, and describes an OpenStack response that provides a solution.

IT Virtualization

The IBM Cloud Pak for AIOps model of a nested virtualization scenario can extend the traditional types of models provided by other solutions.

This model can represent a multi-domain view of the world that links IT, network, storage, applications, and services. In addition, it can incorporate concepts such as OpenStack's Heat Orchestration and Slack collaboration relative to traditional IT resources.

Some of the benefits of this approach are:

- To provide extra context

  Increasingly, teams are more multi-disciplined and no longer operate in informational or functional silos. For example, network teams may include IT Virtualization specialists.

  Such teams need access to more context when needed, to be able to answer some of their business-critical questions, such as:

  - What storage volume is a VM attached to?
  - Which orchestration step realized a network port?
  - Who collaborated with whom for a particular incident?
  - Which applications and services are supported by a network subnet?
  - Which VM instances were shut down as part of a scale-in activity 1 hour ago?
  - What is the impact of removing a Hypervisor from the environment?
  - Which fixed IP addresses have a floating IP address that has been bound to in the last week?
- To provide a data-rich base

  Added value services can be bolted onto a base system, provided the information exists, and the system has an architecture that allows for rapid extension.

  For example, when building analytics on the topology data, the availability of information such as seasonality can provide extra insights.

  The following diagram depicts the nested layers of virtualization, including networking, between these layers and technologies such as Docker and LXC or LXD.

  Note: The services that are exposed can be applications or appear to be traditionally physical services such as network routers, switches, firewalls, and load balancers (a key goal of NFV).

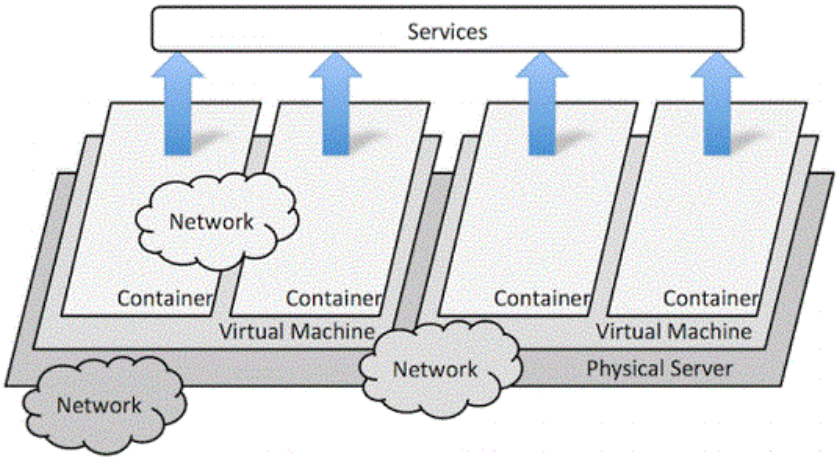


Figure. Physical services

OpenStack

OpenStack is a free and open source platform for cloud computing, typically deployed as an IaaS (Infrastructure-as-a-Service) capability. The software platform consists of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and network resources throughout and between data centers.

OpenStack provides a number of projects, and related services and APIs, that are summarized here, as they speak directly to the need to have a multi-domain view of the environment. For more information, see the OpenStack project navigator

OpenStack **core services** include the following services:

- Nova

  Compute manages the lifecycle of compute instances in an OpenStack environment.

- Neutron

  Networking enables network connectivity as a service for other OpenStack services.

- Swift

  Object Storage stores and retrieves arbitrarily unstructured data through a REST API.

- Cinder

  Block Storage provides persistent storage to running instances.

- Keystone

  Identity provides authentication and authorization services to OpenStack services and a service catalog.

- Glance

  Image Service stores and retrieves virtual machine disk images for use by Nova.

OpenStack **optional services** include the following services:

- Horizon

  dashboarding

- Ceilometer

  telemetry

- Heat

  orchestration

- Sahara

  Platform Symphony MapReduce

- Designate

DNS

- Barbican

  Key Management

IBM Cloud Pak for AIOps provides an Observer that makes extensive use of OpenStack's core and Heat APIs to build an end-to-end (or multi-domain) model.

IT Virtualization OpenStack scenario

The following example of an OpenStack environment accessed through IBM Cloud Pak for AIOps provides insights into any environment consisting of, for example, a combination of physical services and storage combined with virtualization from VMware (or similar).

IT Virtualization OpenStack scenario

The following example of an OpenStack environment provides insights into any environment consisting of, for example, a combination of physical services and storage combined with virtualization from VMware, or similar virtualization providing infrastructure.

IT virtualization

The following figure depicts a Hypervisor running a VM (LB Server) that is assigned to a tenant, has classification data and a number of network interfaces. Each network interface has a MAC address, one or more IP addresses (v4 or v6) and each IP address is considered to be part of an IP subnet. It is also related to orchestration data from Heat that helps identify how the instance was created.

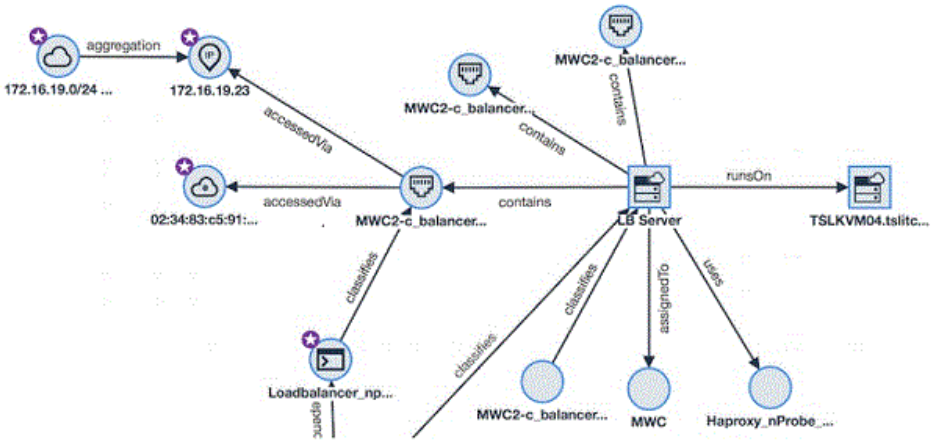


Figure. Example multi-domain topology

Property use guidance

Although the topology service is flexible, you should use the following guidelines when setting property values to ensure that elements are appropriately represented:

- Set **`entityType`** to one of the pre-defined entity types with an associated icon where possible.
- Set **`name`** and **`description`** to values that are easy to read or remember wherever possible.
- Use the default generic properties to represent generally applicable characteristics of an element and to provide a basic degree of labeling and filtering. For a list of generic properties, see the following topic: Properties

Model patterns - Part one

Stepping through each of the sections of the example of a multi-domain topology helps to identify reusable patterns.

In the following figure, the Hypervisor 'TSLKVM04' is running a VM 'LB Server'.

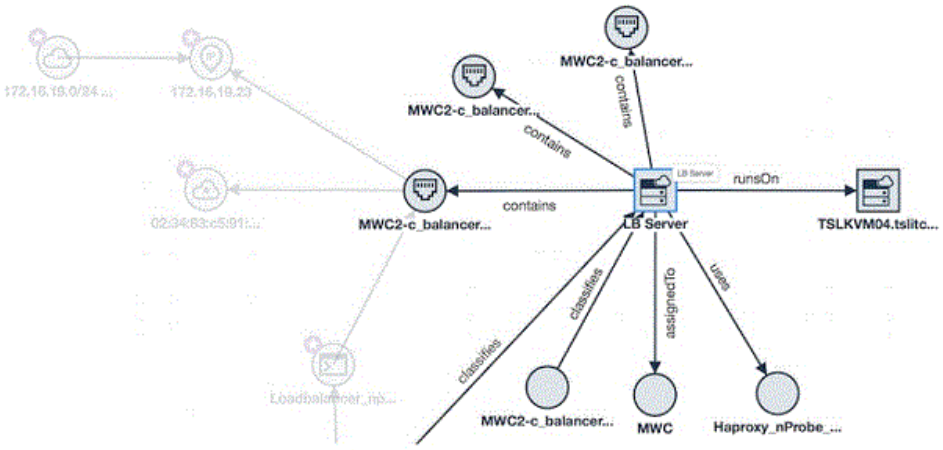


Figure. Figure of Hypervisor

The LB Server in this image is:

- Assigned to the 'MWC' tenant
- Contains three network interfaces
- Uses the Haproxy _nProbe image (OpenStack image)
- Is classified as an MWC2-c _balancer (OpenStack flavor)

This means that the following pattern can be identified:

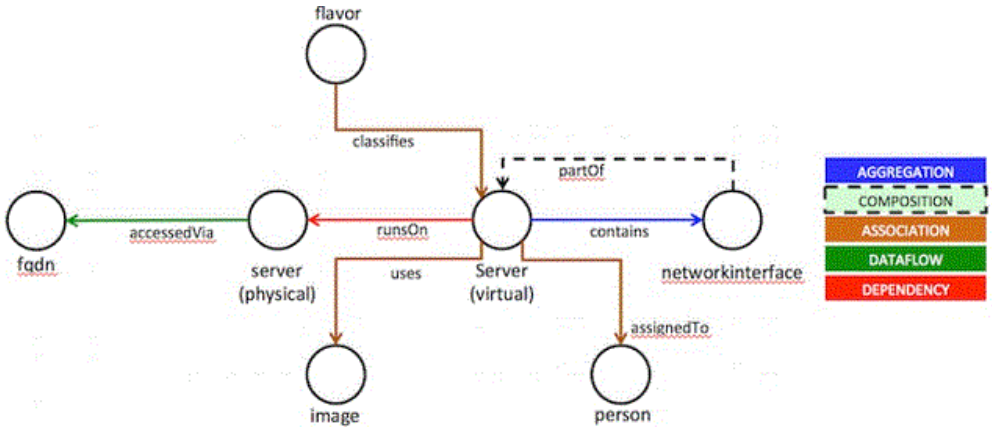


Figure. Pattern

Usage tips:

- The associations shown for flavor, image and person are optional.
- Network interfaces and other components of the device must be associated with it through a partOf relationship. The exception is if an IP or MAC address is known independently of any device, for example flow data would expose those but the device would be unknown.
- The FQDN (hostname, short or full DNS name) is associated directly with the server in this case as nothing else is known about the Hypervisor.

  Note: This is **not** shown in the topology fragment.
- The **`partOf`** composition is not shown by the topology GUI, but must be created where the relationship between a component and a device is known. This is in addition to relationships such as **`contains`**, which the IBM Cloud Pak Automation console will show.

Hypervisor example JSON

```
{
    "_executionTime": 3,
    "createTime": 1501741671066,
    "name": "TSLKVM04",
    "uniqueId": "CYooventry_DC1:MWC/ComputeHost/TSLKVM04",
    "observedTime": 1501776262368,
    "_startedAt": "2017-08-03T06:27:51.066Z",
    "entityTypes": [
        "server"
    ],
    "beginTime": 1501741671066,
    "_id": "KNN6TCGhKyM4MCI6jooGWg",
```

```
    "_observedAt": "2017-08-03T16:04:22.368Z",
    "_modifiedAt": "2017-08-03T06:27:51.066Z",
    "_createdAt": "2017-08-03T06:27:51.066Z",
    "changeTime": 1501741671066,
    "matchTokens": [
        "Coventry_DC1:MWC/ComputeHost/TSLKVM04",
        "TSLKVM04"
    ]
}
```

Note: Many of the properties starting with _ are internal (such as timestamps). Also:

- The `uniqueId` is a composite of a number of fields: data center, tenant, className and name of the Hypervisor because the ID is a highly ambiguous integer.
- The `name` is the name of the instance from OpenStack.
- The `entityType` is set to 'server' to ensure correct classification and icon use.

Virtual machine example JSON

```
{
    "instanceName": "LB server",
    "tenantId": "2f79c691570c4a598be386325ea01da8",
    "launchedAt": 1501741751194,
    "_executionTime": 4,
    "userId": "48c7cd25ad0842be8e9b84390de0e587",
    "imageName": "None Available",
    "availabilityZone": "nova",
    "createTime": 1501741733817,
    "flavorName": "m1.nano",
    "name": "LB server",
    "uniqueId": "1e35c68a-86b0-445f-9741-e581121a0577",
    "serverStatus": "active",
    "observedTime": 1501741752717,
    "_startedAt": "2017-08-03T06:29:12.717Z",
    "entityTypes": [
        "server",
        "vm"
    ],
    "beginTime": 1501741752717,
    "flavorId": "42",
    "vmState": "active",
    "_id": "3nDmTkKfrvNhZinSDCYHDw",
    "_observedAt": "2017-08-03T06:29:12.717Z",
    "createdAt": 1501741733000,
    "_modifiedAt": "2017-08-03T06:29:12.717Z",
    "_createdAt": "2017-08-03T06:28:53.817Z",
    "changeTime": 1501741752717,
    "matchTokens": [
        "1e35c68a-86b0-445f-9741-e581121a0577",
        "LB server"
    ]
}
```

Note: Many of the properties starting with _ are internal (such as timestamps). Also:

- The `uniqueId` in this case is the UUID of the instance from OpenStack.
- The `name` is the name of the instance from OpenStack.
- The `entityType` is set to 'server' and 'vm' to ensure correct classification and icon use.
- The `isVirtual` Boolean is set to true.

Network interface example JSON

```
{
    "_executionTime": 3,
    "isAdminStateUp": true,
    "createTime": 1501741717674,
    "name": "aab47c85-3110-401a-8bad-960b7c4bcd7b",
    "uniqueId": "aab47c85-3110-401a-8bad-960b7c4bcd7b",
    "observedTime": 1501741718855,
    "_startedAt": "2017-08-03T06:28:38.855Z",
    "entityTypes": [
        "networkinterface"
    ],
    "beginTime": 1501741718855,
    "isPortSecurityEnabled": true,
    "_id": "kkKpDH6yLn7cmNVXOddImg",
    "_observedAt": "2017-08-03T06:28:38.855Z",
```

```
    "_modifiedAt": "2017-08-03T06:28:38.855Z",
    "_createdAt": "2017-08-03T06:28:37.674Z",
    "changeTime": 1501741718855,
    "matchTokens": [
        "aab47c85-3110-401a-8bad-960b7c4bcd7b"
    ]
}
```

Note: As with the LB Server data, many of the properties are internal. Also:

- The **uniqueId** in this case is the UUID of the instance from OpenStack.
- The **name** is the name of the instance from OpenStack.
- The **entityType** is set to **networkinterface** to ensure correct classification and icon use.
- The type of the interface (such as **ifType** from IBM Tivoli Network Manager) is unknown.

Model patterns - Part two

Note: Each section of the example of the multi-domain topology identifies the reusable patterns.

In the following figure, the LB Server VM contains a network interface.

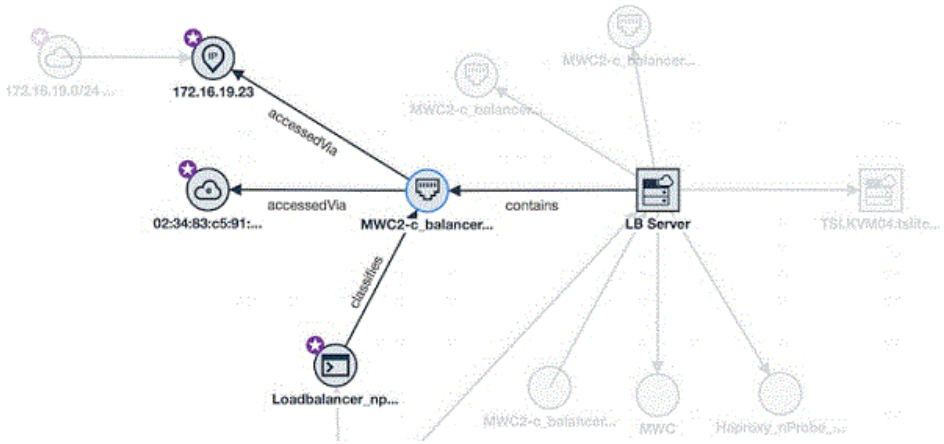


Figure. LB Server VM

The network interface in this image is:

- Accessed through an IP address and a MAC address
- Classified by a Heat orchestration resource

This means that the following pattern can be identified:

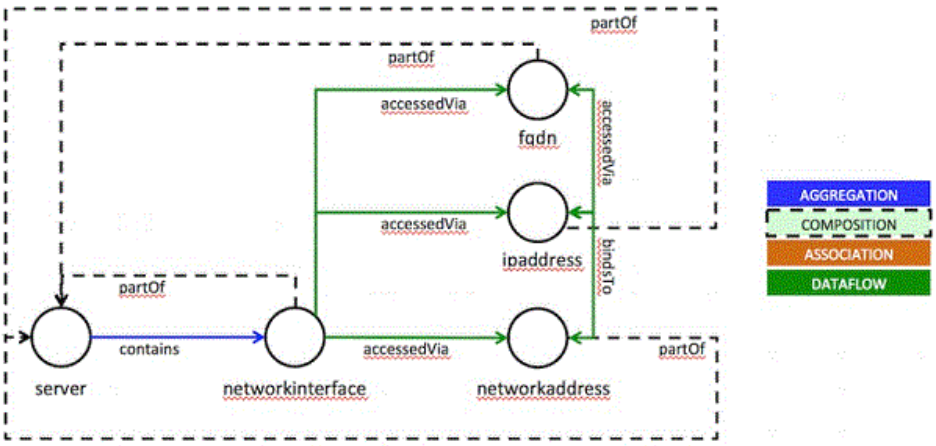


Figure. Pattern

Usage Tips:

- The Heat orchestration element and relationship is optional. Such things can be added if known to provide extra context.
- The network interface must be contained by and **partOf** the device. **Contains** is more fine-grained and may reference intermediate cards (for example) within the containment hierarchy of the device, such as

`json device--contains-->card--->contains--->port`

- If an IP address, FQDN or MAC address is known to be related to a specific device, then they must be associated with the device through a **`partOf`** relationship *in addition* to the **`accessedVia`** relationship.
- If an IP address, FQDN, or MAC address is known independently of a device, then no **`partOf`** relationship from them is necessary.
- If an IP address is known to resolve to an FQDN, then relationships between them should be created to depict that one resolves to another and one accesses another (**`accessedVia`** shown in the example).

IP address example JSON

```
{
    "_executionTime": 3,
    "createTime": 1501741717656,
    "name": "172.24.4.5",
    "uniqueId": "172.24.4.5",
    "ipNumber": 2887255045,
    "addressSpace": "Coventry_DC1:MWC",
    "observedTime": 1501741731565,
    "_startedAt": "2017-08-03T06:28:51.565Z",
    "entityTypes": [
        "ipaddress"
    ],
    "beginTime": 1501741731565,
    "version": "IPv4",
    "_id": "jPLc72DU-UvPeTQE_7YdPQ",
    "_observedAt": "2017-08-03T06:28:51.565Z",
    "_modifiedAt": "2017-08-03T06:28:51.565Z",
    "_createdAt": "2017-08-03T06:28:37.656Z",
    "changeTime": 1501741731565,
    "matchTokens": [
        "172.24.4.5",
        "IPv4:2887255045"
    ]
}
```

Note:

- The **`uniqueId`** is the IP address itself. This is an RFC1918 IP address and so it **must** be qualified with an address space to disambiguate it from other instances of the same IP address. Similar precautions should be used with MAC addresses, which can be ambiguous.
- The **`protocol`** reflects whether the IP address is an IPv4 or IPv6 address.
- The **`ipNumber`** is a numeric representation of the IPv4 or IPv6 address. A Java BigInteger has the precision to represent IPv6 addresses.

Model patterns - Part three

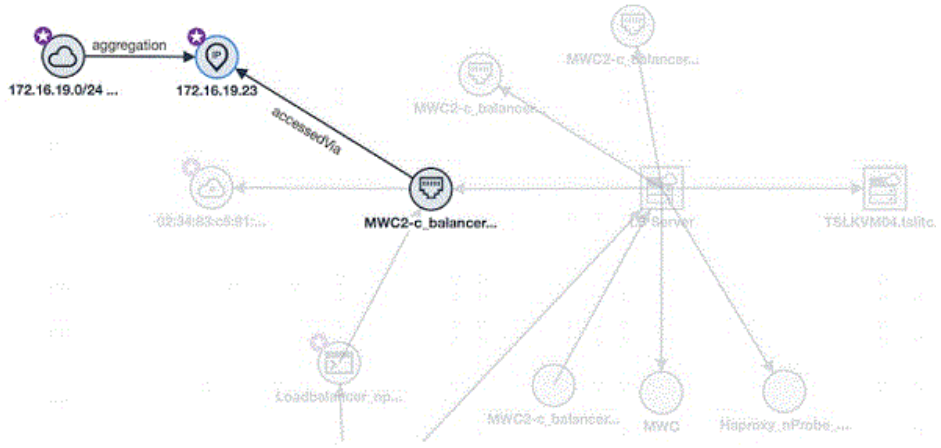


Figure. Multi-domain topology

This means that the following pattern can be identified:

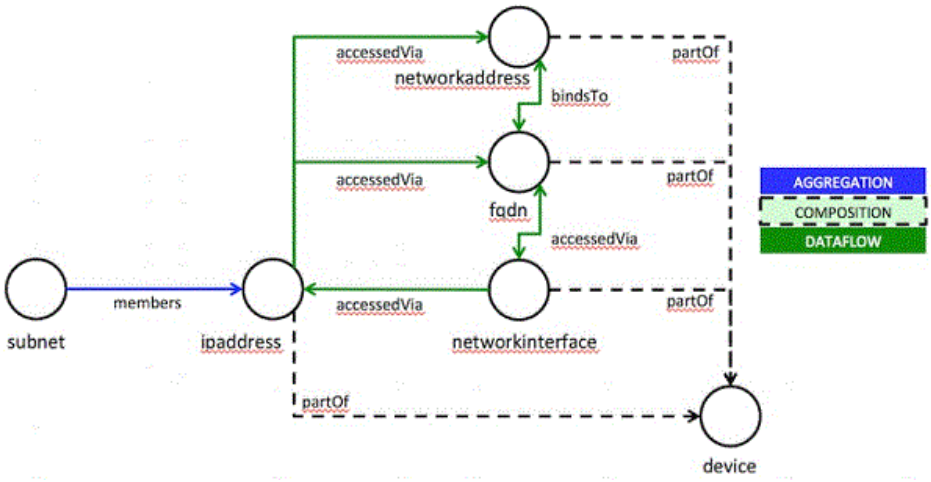


Figure. Pattern

Usage Tips:

- The IP subnet should aggregate any IP addresses known to be in it.
- Elements accessed through an IP should be related to it, such as a network interface, service, or process.
- If an FQDN is known to resolve to an IP address (and vice-versa), then they should be related.
- If a MAC address is known to bind to an IP address and vice-versa, they should be related.
- If an IP address, MAC address, or FQDN is known to relate to a device, they should be considered **`partOf`** it; otherwise, they are independent.

IP subnet example JSON


```
{
  "uniqueId": "c009ff59-13b9-48dc-8863-cd0c75070d99",
  "name": "172.24.4.0/24 (public-subnet)",
  "entityTypes": [
    "subnet"
  ],
  "matchTokens": [
    "172.24.4.0/24 (public-subnet)",
    "172.24.4.0/24",
    "c009ff59-13b9-48dc-8863-cd0c75070d99"
  ],
  "_id": "u0sjaHcumtg5A4DRl1fyAQ",
  "_references": [
    {
      "_id": "9duzx1-3o52g-ys5-47si0",
      "_edgeType": "contains",
      "_label": "aggregation",
      "_fromId": "u0sjaHcumtg5A4DRl1fyAQ",
      "_toId": "lSJQs8JmYzDQ-_wUOfvJbg",
      "_fromUniqueId": "c009ff59-13b9-48dc-8863-cd0c75070d99",
      "_toUniqueId": "172.24.4.12",
      "createTime": 1501838680418,
      "_observedAt": "2017-08-04T09:24:40.418Z",
      "_createdAt": "2017-08-04T09:24:40.418Z",
      "beginTime": 1501838680418,
      "_startedAt": "2017-08-04T09:24:40.418Z",
      "observedTime": 1501838680418
    },
  ],
  "_executionTime": 11,
  "_modifiedAt": "2017-08-04T09:24:40.356Z",
  "isDhcpEnabled": false,
  "dnsNames": "None Available",
  "_observedAt": "2017-08-04T09:24:40.356Z",
  "gatewayIp": "172.24.4.1",
  "_startedAt": "2017-08-04T09:24:40.356Z",
  "observedTime": 1501838680356,
  "changeTime": 1501838680356,
  "ipVersion": "V4",
  "createTime": 1501838680356,
  "_createdAt": "2017-08-04T09:24:40.356Z",
  "cidr": "172.24.4.0/24",
```

```
  "networkId": "3e2b5d07-653a-4fc8-8224-45801d9d113f",
  "beginTime": 1501838680356,
  "allocationPools": "172.24.4.2-to-172.24.4.254"
}
```

Notes:

- The `uniqueId` in this case is the UUID of the subnet from OpenStack.
- The `name` in this case is set to CIDR notation plus the ID of the subnet.
- The `entityType` is set to subnet to ensure appropriate classification and icon usage.
- Some example relationships are shown: For example, an IP address that is part of the subnet is visible, and the subnet's use of an allocation pool is also depicted.

Physical device recipe

The following example of an IBM Tivoli Network Manager environment accessed through IBM Cloud Pak® for AIOps provides insights into any environment that uses physical network devices.

Network physical devices

The following figure depicts two physical devices, 172.20.1.104 and 172.20.1.1, which are connected to each other.

Server 172.20.1.1 has 3 Gigabit Ethernet ports on the same card, one of which has a subinterface with an `ifName` of 'ge-1/1/3.0'. That subinterface shares the MAC address as its physical parent and has two IPv6 addresses and one IPv4 address associated with it.

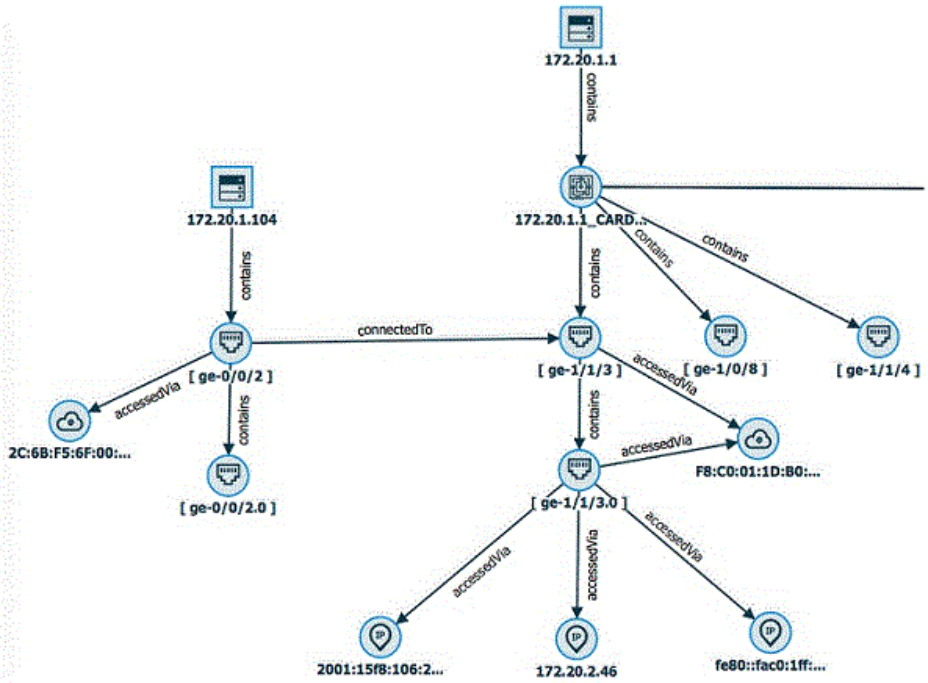


Figure. Figure that shows devices

Outline pattern

The topology service has an edge type of `partOf`, which is a member of the composition edge label family. For more information, see Edge labels.

The following image illustrates how this relationship is used to tie the card, interfaces, and network addresses to the hosting device.

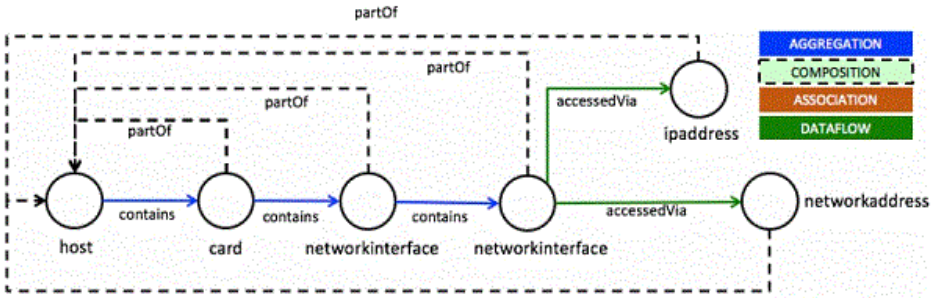
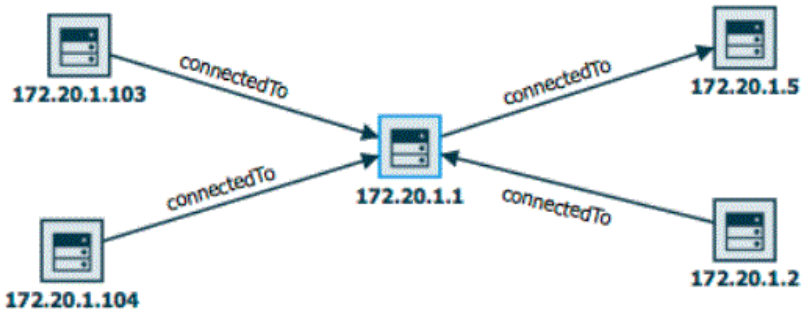


Figure. Relationships to hosting device

The **partOf** relationship is not shown as an explicit edge in the topology UI. However, it is used to determine which resources should be hidden in the host-to-host view, which in the context of this scenario would show just the hosts 172.20.1.104 and 172.20.1.1. It would also show the **connectedTo** edge between them, as depicted in the following part of the image.



{: caption="Figure. Figure that shows **partOf** relationship" caption-side="bottom"}

Host example JSON

The following JSON extract is an example of the properties that you might choose to include when creating a host vertex. The properties that start with an underscore character are aliases for some of the read-only generic properties. They also include a few read/write generic properties, such as **name** and **uniqueId**. However, most of the properties in this example are free from 'User' properties.

```
{
  "uniqueId": "NCOMS:172.20.1.1",
  "name": "172.20.1.1",
  "entityTypes": [
    "host"
  ],
  "_createdAt": "2017-03-03T16:02:40.845Z",
  "_observedAt": "2017-03-03T16:04:18.636Z",
  "_id": "y7EXOKrHud21CWySCyMsBg",
  "_href": "/1.0/topology/resources/y7EXOKrHud21CWySCyMsBg",
  "_nodeType": "resource",
  "_executionTime": 4,
  "_modifiedAt": "2017-03-03T16:02:40.845Z",
  "matchTokens": [
    "sbk-pe1-jrmx80.southbank.eu.test.lab"
  ],
  "sysObjectId": "1.3.6.1.4.1.2636.1.1.1.2.90",
  "entityChangeTime": "2017-03-03T14:13:17.000Z",
  "className": "JuniperMSeries",
  "services": "datalink(2) network(3)",
  "_startedAt": "2017-03-03T16:02:40.845Z",
  "manual": 0,
  "cdmAdminState": 0,
  "cdmType": 2,
  "sysDescription": "Juniper Networks, Inc. mx5-t internet router, kernel JUNOS 15.1F4.15, Build date: 2015-12-23 20:50:37 UTC Copyright (c) 1996-2015 Juniper Networks, Inc.",
  "sysName": "sbk-pe1-jrmx80.southbank.eu.test.lab",
  "sysLocation": "The Mad Hatter Hotel 3-7 Stamford St London SE1 9NY UK,-0.10499474,51.50711477",
  "interfaceCount": 73,
  "entityCreateTime": "2017-03-03T14:13:17.000Z",
  "isIpForwarding": "forwarding",
  "accessIPAddress": "172.20.1.1",
  "entityDiscoveryTime": "2017-03-03T14:11:09.000Z",
  "sysContact": "williamking@uk.ibm.com",
  "_tenantIds": [
    "MoaldcmKHfx3dlyJnGm6JQ"
```

```
  ],
  "accessProtocol": "IPv4"
}
```

Note: Some of the generic properties in this example are:

- The `uniqueId` in this case is a string, which uniquely identifies this resource to IBM Tivoli Network Manager.
- The `name` is the name of the host and will be used in the UI.
- The `entityType` is set to 'host'.
- The `matchTokens` contains the FQDN name of the host.

Network interface example JSON

```
{
  "uniqueId": "NCOMS:172.20.1.1[ ge-1/1/3.0 ]",
  "name": "[ ge-1/1/3.0 ]",
  "entityTypes": [
    "networkinterface"
  ],
  "_createdAt": "2017-03-03T16:03:03.889Z",
  "_observedAt": "2017-03-03T16:03:03.939Z",
  "_id": "v8aFVG6JoigYxTrlFSDkLw",
  "_href": "/1.0/topology/resources/v8aFVG6JoigYxTrlFSDkLw",
  "_nodeType": "resource",
  "_executionTime": 5,
  "operationalStatus": "started",
  "ifIndex": 537,
  "ifAdminStatus": "up",
  "_modifiedAt": "2017-03-03T16:03:03.940Z",
  "ifType": 53,
  "matchTokens": [
    "ge-1/1/3.0",
    "ifEntry.537"
  ],
  "ifName": "ge-1/1/3.0",
  "ifTypeString": "propVirtual",
  "connectorPresent": "false",
  "_startedAt": "2017-03-03T16:03:03.939Z",
  "speed": 1000000000,
  "mtu": 1500,
  "accessIpAddress": "172.20.2.46",
  "operationalDuplex": "FullDuplex",
  "promiscuous": false,
  "physicalAddress": "F8:C0:01:1D:B0:13",
  "ifDescription": "ge-1/1/3.0",
  "ifOperStatus": "up",
  "_tenantIds": [
    "Moa1dcmKHfx3dlyJnGm6JQ"
  ],
  "accessProtocol": "IPv4"
}
```

Note: Some of the generic properties in this example are the following:

- The `uniqueId` in this case is a string, which uniquely identifies this resource to IBM Tivoli Network Manager.
- The `name` is the name of the host and will be used in the UI.
- The `entityType` is set to `networkinterface`.
- The `matchTokens` contains both the `ifName` and a string denoting the `ifIndex` in the `ifEntry` table.

Infrastructure management API Documentation

In this guide you will find a detailed overview of REST API, reference material, appendices, and examples. The guides provide information on how to integrate Infrastructure Automation – Infrastructure Management with external applications. It details the specification of the Infrastructure management REST API, which is implemented as standard REST HTTP requests and responses of content type JSON.

- API Overview
- API Reference
- API Appendices
- API Examples
- Prerequisites

Infrastructure management API Overview

In this guide you will find a detailed overview of Infrastructure Automation – Infrastructure Management REST API.

- REST API Authorization
- Password Updates
- CRUD
- HTTP Basics
- JSON Specification
- Query Specification
- Bulk Querying
- Collection Filtering
- Pagination

Prerequisites for Infrastructure management APIs

Infrastructure management can be installed as either a VM aplliance or as a containerized deployment. To issue API calls to the Infrastructure management that is installed as a containerized deployment, obtain the authorization token and the URL hostname as described below.

Set the values for the following variables:

Variable	Description
IM_USER	IM_USER is the user you want to use. You can use either the default admin user or any user in the user group that you have configured in the im-iminstall CR. See https://ibm.biz/IM_containerized_pod.
IM_PASSWORD	Set the user password variable.
CP_CONSOLE_HOSTNAME	Set the CP console hostname with the command `oc -n ibm-common-services get routes cp-console -o=jsonpath='{.spec.host}'`.

1. Run the command to get the bearer token for the user

   ```
   export IM_BEARER_TOKEN=$(curl -ks -X POST  -H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8"  -d "grant_type=password&username=${IM_USER}&password=${IM_PASSWORD}&scope=openid" https://${CP_CONSOLE_HOSTNAME}/idprovider/v1/auth/identitytoken|jq -r .access_token)
   ```

2. Run the command to get the URL hostname,

   export IM_HOSTNAME=$(oc -n cp4aiops get route | grep inframgmt | awk '{print $2}')

3. After the variables IM_BEARER_TOKEN and IM_HOSTNAME are defined, use them to issue the API calls. A curl command should be in this format,

```
curl  -H "Authorization: Bearer ${IM_BEARER_TOKEN}" \
     ..............................  \
     "https://${IM_HOSTNAME}/api/......"
```

For example, to issue a GET request to the Infrastructure management that is installed as a containerized deployment, specify,

```
curl  -H "Authorization: Bearer ${IM_BEARER_TOKEN}" \
     -k -i -X GET -H "Accept: application/json"   \
     "https://${IM_HOSTNAME}/api/providers"
```

Note: To issue API calls to the Infrastructure management that is installed as a VM appliance, follow the instructions in the subsequent sections in Infrastructure management API Guide. To issue API calls to the Infrastructure management that is installed as a containerized deployment, replace the HTTP authorization header and the URL segment (http://localhost:3000) with those described above.

- REST API Authorization
  - Specifying Authorization Group
  - Specifying an Invalid Authorization Group

REST API Authorization

When authenticating to the appliance, the current group of the authenticated user will be used for determining the authorization of what the user can see and do using the REST API. While this is acceptable when users only belong to a single group, this falls short when users are authenticating using their External Authentication credentials (IPA, LDAP or AD) where they may belong to multiple groups.

Specifying Authorization Group

This section describes how to specify an authorization group for the user. The mechanism described here is honored for both methods of authentication:

- Basic Authentication

- Token based Authentication

The user's group is to be specified in the new HTTP Header:

```
X-MIQ-Group
```

and is to contain the user's group name.

Example request using Basic Authentication:

```
$ curl --user username:password
        -i -X GET -H "Accept: application/json"
        -H "X-MIQ-Group: auditors"
        http://localhost:3000/api/vms
```

Example request using Token based Authentication:

```
$ curl -i -X GET -H "Accept: application/json"
       -H "X-Auth-Token: 93d98fd44cc5a401ff385ba87b21fd13"
       -H "X-MIQ-Group: auditors"
       http://localhost:3000/api/vms
```

The validated users's group will be in effect for the duration of the REST API request. Multiple requests can be made as a particular user but with different authorization groups concurrently.

Specifying an Invalid Authorization Group

In case an invalid authorization group is specified, a 401 Unauthorized is returned:

```
HTTP/1.1 401 Unauthorized
X-Frame-Options: SAMEORIGIN
...

{
  "error" : {
    "kind" : "unauthorized",
    "message" : "Invalid Authorization Group <invalid_group_name> specified",
    "klass" : "ApiController::AuthenticationError"
  }
}
```

- Password Updates

Password Updates

When the appliance is configured for Database authentication, users can update their passwords via the REST API.

Note that if an authenticated REST API user is entitled to edit users, they can edit other users' passwords, e.g. admin account.

Passwords can be updated via the **edit** post action on the user's resource as follows:

```
POST /api/users/:id

{
  "action" : "edit",
  "resource" : { "password" : "<new_password>" }
}
```

So for updating one's password, the */api/users/:id* being updated must be the authenticated REST API user's object.

- CRUD Request Examples
  - Show a collection of Resources
  - Return a single Resource
  - Create a Resource
  - Update a Resource
  - Delete a Resource

CRUD Request Examples

The following examples show the basic CRUD operations (Create, Read, Update, Delete) using the REST API.

For Authentication, we are showing here basic authentication via the *–user admin:smartvm* credentials argument. For multiple API calls it is recommended to access the REST API via token based authentication, please refer the the Authentication Page for details.

Show a collection of Resources

Get a collection of services: **GET /api/services**

```
curl --user admin:smartvm
      -i -X GET -H "Accept: application/json"
      http://localhost:3000/api/services
```

Return a single Resource

Return a single service: **GET /api/services/:id**

```
curl --user admin:smartvm
      -i -X GET -H "Accept: application/json"
      http://localhost:3000/api/services/1
```

Create a Resource

Create a new provider: **POST /api/providers**

```
curl --user admin:smartvm
      -i -X POST -H "Accept: application/json"
      -d '{
            "type"      : "ManageIQ::Providers::Redhat::InfraManager",
            "name"      : "RHEVM Provider",
            "hostname"  : "rhevm.local.com",
            "ipaddress" : "192.168.5.1",
            "credentials" : {
              "userid"   : "admin",
              "password" : "12345"
            }
      }'
      http://localhost:3000/api/providers
```

Update a Resource

Update the name of a service: **PUT /api/services/:id**

```
curl --user admin:smartvm
      -i -X PUT -H "Accept: application/json"
      -d '{ "name" : "updated service name" }'
      http://localhost:3000/api/services/1
```

Delete a Resource

Delete a service: **DELETE /api/services/:id**

```
curl --user admin:smartvm
      -i -X DELETE -H "Accept: application/json"
      http://localhost:3000/api/services/1
```


- HTTP Basics
- REST API Entry Point
  - Response:
  - Entry Point Details
- Supported Content Types
- URL Paths
- Methods and related URLs
- Updating resources
- Modifying Resource attributes
- Return Codes
  - Success
  - Client Errors
  - Server Errors

HTTP Basics

- REST Entry Point
  - Entry Point Details
- Supported Content Types
- URL Paths
- Methods and related URLs
- Updating resources

- Return Codes

REST API Entry Point

The REST API is made available via the /api URL prefix. It can be accessed on the Appliance server directly via HTTP using the default port as follows:

```
http://localhost:3000/api
```

Or external to the Appliance as follows:

```
https://<applianceHost_fqdn>/api
```

Response:

```
{
  "name" : "API",
  "description" : "REST API",
  "version" : "2.3.0",
  "versions" : [
    {
      "name" : "2.3.0",
      "href" : "http://localhost:3000/api/v2.3.0"
    }
  ],
  ...
  "collections" : [
    ...
    {
      "name" : "providers",
      "href" : "http://localhost:3000/api/providers",
      "description" : "Providers"
    },
    ...
    {
      "name" : "vms",
      "href" : "http://localhost:3000/api/vms",
      "description" : "Virtual Machines"
    },
    ...
  ]
}
```

- Version is the Current API version, accessible via either of the following:
- /api/
- /api/v2.0.0/
- Versions lists current and any earlier API versions that are still exposed via their respective entry points:
- /api/v**Version**/

Entry Point Details

An authenticated request of the entry point:

```
GET /api
```

returns the following structure:

```
{
  "name": "API",
  "description": "REST API",
  "version": "2.3.0",
  "versions": [
    {
      "name": "2.3.0",
      "href": "http://localhost:3000/api/v2.3.0"
    }
  ],
  "settings": {
  },
  "identity": {
    "userid": "admin",
    "name": "Administrator",
    "user_href": "http://localhost:3000/api/users/1",
    "group": "EvmGroup-super_administrator",
    "group_href": "http://localhost:3000/api/groups/2",
```

```
    "role": "EvmRole-super_administrator",
    "role_href": "http://localhost:3000/api/roles/1",
    "tenant": "My Company",
    "groups": [
      "EvmGroup-super_administrator"
    ]
  },
  "server_info": {
    "version": "master",
    "build": "20161104133757_01d7244",
    "appliance": "EVM"
  },
  "product_info": {
    "name": "ManageIQ",
    "name_full": "ManageIQ",
    "copyright": "Copyright (c) 2014-present ManageIQ Authors.",
    "support_website": "http://www.manageiq.org",
    "support_website_text": "ManageIQ.org"
  },
  "collections": [
    {
      "name": "actions",
      "href": "http://localhost:3000/api/actions",
      "description": "Actions"
    },
    ...
    {
      "name": "vms",
      "href": "http://localhost:3000/api/vms",
      "description": "Virtual Machines"
    },
    {
      "name": "zones",
      "href": "http://localhost:3000/api/zones",
      "description": "Zones"
    }
  ]
}
```

Where **settings** and **identity** are specific to the authenticated user.

Supported Content Types

Requests:

```
Accept: application/json
```

Responses:

```
Content-Type: application/json
```

URL Paths

The recommended convention for URLs is to use alternate collection / resource path segments, relative to the API entry point as described in the following example:

URL	Description
/api	The REST API entry point
/api/v**Version**	The REST entry point for a specific version of the REST API
/api/:collection	A top-level collection
/api/:collection/:id	A specific resource of that collection
/api/:collection/:id/:subcollection	Sub-collection under the specific resource

Methods and related URLs

The basic HTTP Methods used for the API are GET, POST, PUT, PATCH, OPTIONS and DELETE.

URL	Semantic
GET /api/:collection	Return all resources of the collection
GET /api/:collection/:id	Return the specific resource
POST /api/:collection	Create a resource in the collection
POST /api/:collection/:id	Perform an Action on a resource in the collection

URL	Semantic
PUT /api/:collection/:id	Update a specific resource
PATCH /api/:collection/:id	Update a specific resource
OPTIONS /api/:collection	Return metadata about the collection Please refer to Collection Metadata for details
DELETE /api/:collection/:id	Delete a specific resource

There :collection represent specific Appliance entities like services, hosts, vms, etc.

Updating resources

As shown in the above table, there are a couple of way to update attributes in a resource. These are:

- Update a resource via the PUT HTTP Method
- Update a resource via a POST Method with an *edit* action.
- Update a resource via the PATCH HTTP Method

While the PUT is the common method, the PATCH mechanism gives better control on which attribute to edit, add as well as remove which is not available with the other two methods.

Modifying Resource attributes

```
PUT /api/services/42

{
  "name" : "The new Service name",
  "description" : "A Description for the new Service"
}

POST /api/services/42

{
  "action" : "edit",
  "resource" : {
    "name" : "A new Service name",
    "description" : "A Description for the new Service"
  }
}

PATCH /api/services/42

[
  { "action": "edit", "path": "name", "value": "A new Service name" },
  { "action": "add", "path": "description", "value": "A Description for the new Service" },
  { "action": "remove", "path": "display" }
]
```

Return Codes

- Success
- Client Errors
- Server Errors

Success

- **200 OK** - The request has succeeded without errors, this code should be returned for example when retrieving a collection or a single resource.
- **201 Created** - The request has been fulfilled and resulted in a **new resource being created**. The resource is available before this status code is returned. The response includes the HTTP body of the newly created resource.
- **202 Accepted** - The request has been accepted for processing, but the processing has not been completed. Like, resource is not fully available yet. This status code is usually returned when the resource creation happens asynchronously. In this case the HTTP response includes a pointer to *monitor* or a *job* where the client can query to get the current status of the request and the estimate on when the request will be actually fulfilled.
- **204 No Content** - The server has fulfilled the request but does not need to return an entity-body, and might want to return updated meta information. This HTTP response is commonly used for the DELETE requests, as the resource that was deleted does not exists anymore.

Client Errors

- **400 Bad Request** - The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications. In REST API this status code should be returned to client when the client use the wrong combination of attributes, like expanding the non-existing collection, or using the pagination parameter incorrectly. Another use-case could be creating or performing actions on the resource, when the wrong JSON serialization of the resource or action is used.
- **401 Unauthorized** - The request requires user authentication. The response MUST include a *Authenticate* header field containing a challenge applicable to the requested resource. If the request include *Authenticate* header, then this HTTP status code might indicate that the current user is **not authorized** to perform given action or to access given resource.
- **403 Forbidden** - The server understood the request, but is refusing to fulfill it. Authorization will not help in this case. This HTTP status code might indicate that the action performed is not supported for this resource or collection.
- **404 Not Found** - In this case, the server has not found anything that matches with the URL.
- **415 Unsupported Media Type** - The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method. This error must be returned, when the client is explicitly asking for format other than JSON (application/json).

Server Errors

- **500 Internal Server Error** - The server encountered an unexpected condition which prevented it from fulfilling the request. This error code must be used when an exception is raised in the application and the exception has nothing to do with the client request.

- JSON Specification
  - General Conventions
  - Basic types
  - Common Attributes and Actions
    - Attributes
  - Actions
    - About permissions and security:
  - Collections
  - Action Specification
  - Collection actions
  - Resource actions
  - Action Responses

JSON Specification

General Conventions

The API uses JSON throughout; the Content-Type for all requests and responses is **application/json**.

As is general practice with REST, clients should not make assumptions about the server's URL space. Clients are expected to discover all URL's by navigating the API. To keep this document readable, we still mention specific URL's, generally in the form of an absolute path. Clients should not use these, or assume that the actual URL structure follows these examples, and instead use discovered URL's. Any client should start its discovery with the API entry point, here denoted with **/api**.

- Basic Types
- Common Attributes and Actions
  - Attributes
- Actions
- Collections
- Action Specification
- Collection Actions
- Resource Actions
- Action Responses

Basic types

The following are basic data types and type combinators that are used throughout:

Name	Explanation	Example serialization

Name	Explanation	Example serialization
Integer	Integer value	{ “memsize” : 2048 }
String	JSON string	{ “state” : “running” }
URL	Absolute URL	{ “href” : “http://SERVER/vms/1/start” }
Timestamp	Timestamp in ISO8601 format	{ “created” : “2013-12-05T08:15:30Z” }
Array[T]	Array where each entry has type T	{ “vms” : [{ “id” : “1” }, { “id” : “2” }] }
Ref[T]	A reference to a T, used to model relations, the T is a valid Resource identifier	{ “vm” : { “href” : URL } }
Collection	Array[T] where T represents a Ref[T], this might allow actions to be executed on all members as a single unit	{ “vms” : { “count” : 2, “resources” : [{ “href” : URL}, { “href” : URL }], “actions” : [] }
Struct	A structure with sub-attributes	“power_state”: {“state” : “ON”, “last_boot_time” : “2013-05-29T15:28Z”, “state_change_time” : “2013-05-29T15:28Z”}

Common Attributes and Actions

The following describes attributes and actions that are shared by all resources and collections defined in this API.

Attributes

Attribute	Type	Description
id	String	A string identifier for the referenced resource
href	Ref(self)	A unique self reference
name	String	A human name of the resource

```
{
  "id" : "10",
  "href" : "http://localhost:3000/api/vms/10",
  "name" : "first_vm"
}
```

Actions

Action	HTTP method	Description
create	POST	Create new resource in the collection
edit	PUT/PATCH/POST	Edit attributes in resource
delete	DELETE	Delete resource

Note:

About permissions and security:

Advertising of the common actions depends purely on the role and permissions of that the current API user does have for the particular resource.

Collections

Resources can be grouped into collections. Each collection is homogeneous so that it contains only one type of resource, and is unordered. Resources can also exist outside any collection. In this case, we refer to these resources as singleton resources. Collections are themselves resources as well.

Collections can exist globally, at the top level of an API, but can also be contained inside a single resource. In the latter case, we refer to these collections as sub-collections. Sub-collections are usually used to express some kind of “contained in” relationship

Collections are serialized in JSON in the following way:

```
{
  "name" : String,
  "count": Integer,
  "subcount": Integer,
  "resources": [ ... ],
  "actions": [ ... ]
}
```

Where the **name** attribute is the collection name. The **count** attribute in a collection always denotes the total number of items in the collection, not the number of items returned. **subcount** attribute in a collection depicts the number of items returned. Then the **resources** attribute is an Array[T] where T might be a list of references to the T or, if expanded a list of resources with all attributes. The **actions** then contains an Array of actions that can be performed against the collection resources.

Action Specification

The representation of each resource will only contain an action and its URL if the current user is presently allowed to perform that action against that resource. Actions will be contained in the **actions** attribute of a resource; that attribute contains an array of action definition, where each

action definition has a rel, method and a href attribute.

- **name** attribute contains the action name
- **method** attribute states the HTTP method that must be used in a client HTTP request in order to perform the given action (eg. GET, POST, PUT, DELETE)
- **href** attribute contains the absolute URL that the HTTP request should be performed against
- **form** an optional attribute that references a JSON document which describes the resource attributes that can be provided in the message body when performing this action. This description will indicate which of those attributes are mandatory and which are optional.

Collection actions

The actions performed against a collection of resources are in most cases batch operations against multiple resources. The action request must include an HTTP body with the action name and the list of resource representations that the action will be performed against.

The resource representation might include the resource attributes as they can change the way how the action is actually performed. In the example below, the first service is retired immediately, versus the second being retired at a later date with a retirement warning of 3 days.

Sample JSON request body for collection action:

```
POST /api/services

{
  "action": "request_retire",
  "resources" : [
    { "href" : "http://localhost:3000/api/services/101" },
    { "href" : "http://localhost:3000/api/services/102",
      "date" : "10/30/2015",
      "warn" : 3
    }
  ]
}
```

Actions in collection:

```
{
  "name" : String,
  "count": Integer,
  "resources": [ ... ],
  "actions": [
    {
      "name"   : "start",
      "method" : "post",
      "href"   : URL
    },
    {
      "name"   : "stop",
      "method" : "post",
      "href"   : URL
    },
    {
      "name"   : "suspend",
      "method" : "post",
      "href"   : URL
    },
    {
      "name"    : "edit",
      "method" : "post",
      "href"   : URL
    },
    {
      "name"    : "delete",
      "method" : "post",
      "href"   : URL
    },
    {
      "name"   : "delete",
      "method" : "delete",
      "href"   : URL
    }
  ]
}
```

Resource actions

An action performed against a given resource is always described in the body of the HTTP request. The HTTP body could contain a list of resource attributes that dictate how the state of the receiving resource is to be changed once the action is performed. At minimum the JSON document in

the message body must contain the name of the action to be performed.

In cases where no attributes are required to perform an action the HTTP body will contain an empty JSON document, in which case default values will be assigned to the corresponding attributes.

Sample JSON request body for resource action:

```
POST /api/services/101

{
  "action"   : "request_retire",
  "resource" : { "date" : "10/30/2015", "warn" : 5 }
}

POST /api/vms/321

{
  "action"   : "start",
  "resource" : {}
}
```

or Simply:

```
{
  "action"   : "start"
}
```

Actions in a resource:

```
{
  "id"     : String,
  "href"   : Ref(self),
  "name"   : "resource human name",
  "actions" : [
    {
      "name"   : "edit",
      "method" : "post",
      "href"   : URL
    }
  ]
}
```

Action Responses

When performing actions on resources, there are two types of responses that one is to expect.

1. For actions that operate on the resource itself like a *create* or *edit*, the response is usually the updated resource. This includes creation of Provision or Automate requests where the created /api/provision_requests and /api/automation_requests gets returned.

2. For others like a *start* or *stop* action, the response includes an action result for each targetted resource. An action result has the following construct in the response:

```
{
  "results" : [
    {
      "success" : true | false,
      "message" : String,
      "href" : Ref[resource],
      "result" : Struct,
      "task_id" : Id,
      "task_href" : Ref[task]
    },
    ...
  ]
}
```

results being an array of action results as one or more resources could be targeted in a request.

NOTE:

- success and message are always there.
- result is optional and would exist when an action results in data, i.e. policy resolve.
- href is populated for the resource being targeted by the action
- task_id and task_href are optional. They are defined when actions are run asynchronously and a task is created, i.e. http://localhost:3000/api/tasks/:id which can be monitored for action completion.

Other action specific attributes could also be returned in the the action result.

For tagging actions:

```
{
  "tag_category" : String,
  "tag_name" : String,
  "tag_href" : Ref[tags]
}
```

When executing actions on subcollections, like *policies*, *policy_profiles* and *service_templates*, the following is also provided in the action result:

```
{
  "<subcollection>_id" : Id,
  "<subcollection>_href" : Ref[subcollection]
}
```


- Query Specification
  - Control Attributes
  - Filtering
  - Expanding Collection
  - Virtual Attributes and Relationships

Query Specification

This specification identifies the controls available when querying collections.

The controls are specified in the GET URL as attribute value pairs as follows:

```
GET /api/:collection?attr1=val1&attr2=val2&...
```

Control Attributes

Category	Attribute	Semantics
Pagination		
	offset	0-based offset of first item to return
	limit	number of items to return. If 0 is specified then the remaining items are returned
	Please refer to Pagination for details on default pagination and pagination links provided by the API	
Scope		
	filter[]	One or more filters to search on. See Filtering below.
	attributes=atr1,atr2,...	Which attributes in addition to id and href to return. If not specified or *all* (default is attributes=*all*), then all attributes are returned
	expand=resources	To expand the resources returned in the collection and not just the href. See Expanding Collection below
Sorting		
	sort_by=atr1,atr2,...	By which attribute(s) to sort the result by. SQL friendly virtual attributes are also supported.
	sort_order=asc or desc	Order of the sort
	sort_options=ignore_case	Option for case insensitive sort

- The **count** attribute in a collection always denotes the total number of items in the collection, not the number of items returned.
- The **subcount** attribute in a collection denotes the number of items from the collection that were returned. For example, as a result of a paged request.

Filtering

GET requests against collections support the following query parameters to enable filtering:

- **filter**: The filter to use for querying the collection, i.e. filter[]=name='myservice%'.

```
GET /api/:collection?filter[]=name='myservice%'
```

For more information on filter[] please reference the Filtering page

Expanding Collection

While in the JSON serialization example the description says that the resource might be a list of references to these resource, using the **expand** parameter they can be expanded to return a full JSON serialization of the resource instead:

GET */api/vms*

```
{
  "name" : "vms",
  "count" : 2,
  "subcount" : 2,
  "resources" : [
    { "href" : "http://localhost:3000/api/vms/1" },
    { "href" : "http://localhost:3000/api/vms/2" }
  ],
  "actions" : []
}
```

GET */api/vms?expand=resources*

```
{
  "name" : "vms",
  "count" : 2,
  "subcount" : 2,
  "resources" : [
    {
      "id" : "1",
      "href" : "http://localhost:3000/api/vms/1",
      "name" : "My First VM",
      ...
    },
    {
      "id" : "2",
      "href" : "http://localhost:3000/api/vms/2",
      "name" : "My Second VM",
      ...
    }
  ],
  "actions" : []
}
```

Virtual Attributes and Relationships

When querying resources, virtual attributes (not database columns) as well as relationships can be queried via the attributes parameter.

For example, while accounts and software are defined subcollections of vms

```
GET /api/vms/166?expand=software&attributes=ipaddresses,lans,storage
```

```
{
  "href": "http://localhost:3000/api/vms/166",
  "id": "166",
  "vendor": "vmware",
  "name": "Dev Nightly Appl",
  "location": "Nightly Appl 201.vmx",
  "host_id": "4",
  "created_on": "2014-11-20T19:37:28Z",
  "updated_on": "2015-03-07T04:32:33Z",
  "storage_id": "12",
  "guid": "a9409862-70ec-11e4-90c6-b8e85646e742",
  "uid_ems": "422f50c6-2ba6-1059-338d-423cc3daf5b4",
  "tools_status": "toolsNotRunning",
  "standby_action": "checkpoint",
  "power_state": "off",
  "state_changed_on": "2014-11-20T19:37:28Z",
  "connection_state": "connected",
  "memory_reserve": 0,
  "memory_reserve_expand": false,
  "memory_limit": -1,
  "memory_shares": 40960,
  "memory_shares_level": "normal",
  "raw_power_state": "poweredOff",
  ...
  "ipaddresses": [
    "192.168.100.1"
  ],
  "lans": [
    {
      "id": "8",
      "switch_id": "6",
      "name": "VM Network",
      "tag": "0",
      "created_on": "2014-11-20T19:37:23Z",
      "updated_on": "2014-11-20T19:37:23Z",
      "uid_ems": "VM Network",
      "computed_allow_promiscuous": false,
      "computed_forged_transmits": true,
      "computed_mac_changes": true
```

```
    }
  ],
  "storage": {
    "id": "12",
    "name": "StarM1-Dev",
    "store_type": "VMFS",
    "total_space": 2134061875200,
    "free_space": 385020329984,
    "created_on": "2014-11-20T19:37:22Z",
    "updated_on": "2015-03-09T13:36:05Z",
    "multiplehostaccess": 0,
    "location": "4e43dd32-c6b7543a-32bf-0010187f038c",
    "uncommitted": 845539212800,
    "ems_ref_obj": "--- !ruby/string:VimString\nstr: datastore-15624\nxsiType:
:ManagedObjectReference\nvimType: :Datastore\n",
    "directory_hierarchy_supported": true,
    "thin_provisioning_supported": true,
    "raw_disk_mappings_supported": true,
    "master": false,
    "ems_ref": "datastore-15624"
  }
  "software": [
    {
      "href": "http://localhost:3000/api/vms/320/software/1",
      "id": "1",
      "name": "OpenOffice",
      "vendor": "OpenOffice.org",
      "vm_or_template_id": "166"
    }
  ]
}
```

As another example, one can query good details on hosts:

```
GET /api/hosts/8?attributes=custom_attributes,ext_management_system,resource_pools,storages,vms,hardware
```

of course, one needs to be careful with queries like these as list of vms for a host could be quite large.

Virtual attributes can also be queried from one-to-one relationships via the dot notation as follows:

```
GET /api/hosts/8?attributes=ext_management_system.id,ext_management_system.guid,ext_management_system.name
```

```
{
  "href": "http://localhost:3000/api/hosts/8",
  "id": "8",
  "name": "test1.sample.com",
  "hostname": "test1.sample.com",
  "ipaddress": "test1.sample.com",
  "vmm_vendor": "vmware",
  "vmm_version": "5.0.0",
  "vmm_product": "ESXi",
  "vmm_buildnumber": "515841",
  ...
  "ext_management_system": {
    "name": "vcenter50",
    "guid": "e84e8c58-bdbd-11e4-8983-b8e85646e742",
    "id": "6"
  }
}
```

With attributes, database attributes, virtual attributes, and relationships can be specified together as in the following example:

```
GET /api/vms/166?attributes=name,raw_power_state,ipaddresses,storage.name
```

```
{
  "href": "http://localhost:3000/api/vms/166",
  "id": "166",
  "name": "Dev Nightly Appl",
  "raw_power_state": "poweredOff",
  "ipaddresses": [
    "192.168.253.1"
  ],
  "storage": {
    "name": "StarM1-Dev"
  }
}
```

This is helpful when specific information is needed out of resources and helps with response time when querying large number of resources as in the following example:

```
GET /api/vms?limit=1000&offset=1000&expand=resources&
    attributes=name,raw_power_state,ipaddresses,storage.name
```

- Bulk Querying

Bulk Querying

While the capability to fetch an individual resource via /api/:collection/:id is available, fetching multiple resources via a filtered search is possible but has limitation on GET url limits and is cumbersome on the filtering syntax.

The Bulk Queries provides this capability by implementing a *query* POST action on the primary collections.

```
POST /api/vms
```

```
{
  "action" : "query",
  "resources" : [
    { "guid" : "...." },
    { "guid" : "...." },
    ...
  ]
}
```

The *query* interface is flexible in that it allows resources to be specified by different relevant attributes. These include:

Identifying Attributes
href
id
guid (*if applicable*)

Some other attributes can also be used as identifying attributes as shown in the following table:

Collection	Optional Identifying Attributes
/api/groups	description
/api/roles	name
/api/tags	name
/api/users	name, userid

As with regular GETs, queries return expanded resources, but also support other query parameters as in the following example:

```
POST /api/vms?attributes=name,raw_power_state
```

```
{
  "action" : "query",
  "resources" : [
    { "id" : 10 },
    { "id" : 20 },
    { "id" : 23 },
    { "id" : 28 },
    { "id" : 30 }
  ]
}
```

Sample response:

```
{
  "results": [
    {
      "href": "http://localhost:3000/api/vms/10",
      "id": "10",
      "name": "test_vm_1",
      "raw_power_state": "down"
    },
    {
      "href": "http://localhost:3000/api/vms/20",
      "id": "20",
      "name": "demo_server_A",
      "raw_power_state": "down"
    },
    {
      "href": "http://localhost:3000/api/vms/23",
      "id": "23",
      "name": "demo_server_B",
      "raw_power_state": "down"
    },
    {
      "href": "http://localhost:3000/api/vms/28",
      "id": "28",
```

```
      "name": "aab_dev_vm",
      "raw_power_state": "up"
    },
    {
      "href": "http://localhost:3000/api/vms/30",
      "id": "30",
      "name": "ipa_dev",
      "raw_power_state": "down"
    }
  ]
}
```

- Collection Filtering

Collection Filtering

When querying collections, filters can be specified by providing one or more filter[]=... parameter.

```
GET /api/:collection?filter[]=...&filter[]=...&filter[]=...
```

Syntax for a filter is as follows:

```
filter[]=attribute op value
```

By default, filters are logically *anded* together, for a logical or, prefix the attribute with *or* as follows:

```
filter[]=or attribute op value
```

Where op supports the following operators:

Operator	Description	Operands
=	Equality	Numeric or string. For String, supports % or * as wildcard NULL or nil can also be specified instead of quoted string
!=	Inequality	
<	Less Than	Numeric
<=	Less Than or Equal	Numeric
>=	Greater Than or Equal	Numeric
>	Greater Than	Numeric

When strings, value to be quoted in single or double quotes. When datetimes, only strict comparison operators are supported.

Example Query of VMs named sample*, created_on > 2019-09-01, and return name and vendor

```
GET /api/vms?expand=resources&attributes=name,vendor&filter[]=created_on>2019-09-01&filter[]=name='sample%'
```

Example Query looking for services that are retired but have an unspecified service_id

```
GET /api/services?expand=resources&attributes=retired&filter[]=service_id=nil&filter[]=retired=true
```

Example Query of VMs with power_state of on or off:

```
GET /api/vms?expand=resources&filter[]=power_state=[on,off]
```

Alternately, you can use the following more verbose syntax using OR:

```
GET /api/vms?expand=resources&filter[]=power_state=off&filter[]=or%20power_state=on
```

Example Query of VMs with more than 4 CPU sockets or more than 16,000 MB of RAM (OR):

```
GET /api/vms?expand=resources&filter[]=num_cpu>4&filter[]=or%20ram_size>16000
```

Example Query of VMs without a specific value (attribute negation):

```
GET /api/vms?expand=resources&filter[]=power_state!=unknown
```

Example Query of VMs where a field is NOT NULL (vms that are either retired or not):

```
GET /api/vms?expand=resources&filter[]=retired!=nil
GET /api/vms?expand=resources&filter[]=retired!=NULL
GET /api/vms?expand=resources&filter[]=retired!=null
```

Example Query of VMs using a boolean:

```
GET /api/vms?expand=resources&filter[]=retired=true
```

Example Query of VMs created before a specific time:

```
GET /api/vms?expand=resources&filter[]=created_on<2023-01-01
```

Attributes specified via the filter[] parameter can include virtual attributes, including one level association of the virtual attribute as follows:

The HTTP **OPTIONS** request can be used on a collection to display all of the collection specific metadata, including the available virtual attributes. For more information please refer to the Collection Metadata page.

```
GET /api/vms?attributes=name,vendor,hardware&filter[]=vendor='vmware'&filter[]=hardware.memory_mb>=8192
```

Here filtering on the virtual attribute *hardware* including the one level association *memory_mb* off hardware.

- Pagination
  - About max_results_per_page

Pagination

Pagination is enabled by default. When querying collections and the *limit* query parameter is not specified, only 1000 resources are returned by default. This is driven by the new advanced setting:

```
:api:
  :max_results_per_page: 1000
```

Example GET with offset and limit specified.

```
GET /api/vms?offset=2&limit=2&sort_by=name&expand=resources
```

```
{
  "name"      : "vms",
  "count"     : 7,
  "subcount"  : 2,
  "pages"     : 4,
  "resources" : [
    {
      "href"   : "http://localhost:3000/api/vms/7254",
      "id"     :  "7254",
      "vendor" : "vmware",
      "name"   : "test-vm1",
      ...
    },
    {
      "href"   : "http://localhost:3000/api/vms/7257",
      "id"     : "7257",
      "vendor" : "vmware",
      "name"   : "test-vm2",
      ...
    }
  ],
 "actions" : [
    ...
  ],
 "links" : {
   "self"     : "http://localhost:3000/api/vms?offset=2&limit=2&sort_by=name&expand=resources",
   "next"     : "http://localhost:3000/api/vms?offset=4&limit=2&sort_by=name&expand=resources",
   "previous" : "http://localhost:3000/api/vms?offset=0&limit=2&sort_by=name&expand=resources",
   "first"    : "http://localhost:3000/api/vms?offset=0&limit=2&sort_by=name&expand=resources",
   "last"     : "http://localhost:3000/api/vms?offset=6&limit=2&sort_by=name&expand=resources"
 }
}
```

With pagination enabled by default, collection queries now return the following:

- “pages” metadata which is the total number of pages for the *limit* specified (or the default 1000).
- “links” metadata section which includes the necessary URL’s for fetching the current page, next and previous pages, as well as the first and last pages.

Notes:

- When the page being returned is the first page (i.e. offset = 0), the *previous* link is not returned.
- When the page being returned is the last page, the *next* page link is not returned.
- For small result sets where *limit* is less than max_results_per_page, the *limit* will be honored.
- For large result sets where the *limit* is greater than max_results_per_page, *limit* will be ignored and the result will be limited to the first max_results_per_page resources.

About max_results_per_page

If the default value for max_results_per_page does not meet your needs, it can be changed using the Advanced Settings feature in the UI. However, it is *NOT* recommended to set it to some arbitrarily large number in an effort to circumvent the pagination feature. The pagination feature is designed to prevent excessive load on the API workers, which is easy to do with large collections such as metric rollups and event streams that may have millions of records.

Infrastructure management API Reference

In this guide you will find a detailed reference for the Infrastructure Automation – Infrastructure Management REST API.

- Primary Collections
- Actions
- Alert Management
- Authentication Management
- Automate Model
- Automate Domains
- Automation Requests
- Automate Workspaces
- Categories and Tags
- Chargebacks, Rates and Assignments
- Cloud Volumes
- Conditions
- Configuration Script Source Management
- Conversion Hosts
- Custom Actions
- Custom Attributes
- Custom Button Management
- Event Streams
- Flavor Management
- Generic Object Management
- Group Management
- Host Management
- Instance Management
- Metric Rollups
- Network Routers
- Notifications Support
- Orchestration Support
- Physical Server Management
- Picture Management
- Policy Management
- Policy Collections
- Provider Support
- Provision Requests
- Request Support
- Reports Support
- Role Management
- Security Groups
- Service Management
- Service Queries
- Service Custom Actions
- Service Dialogs
- Service Orders
- Setting Ownership
- Settings Management
- Snapshot Management
- Tags and Tagging
- Tasks
- Tenant Management
- Transformation Mappings
- User Management
- Virtual Machine Management

- Collections

Collections

The following tables shows the currently available primary collections accessible via:

`GET /api/:collection`

Where :collection is one of the following:

Collection	Description
actions	Actions
alert_definition_profiles	Alert Definition Profiles
alert_definitions	Alert Definitions
alerts	Alerts
auth_key_pairs	Auth Key Pairs
authentications	Authentications
automate	Automate Model
automate_classes	Automate Classes
automate_domains	Automate Domains
automate_workspaces	Automate Workspaces
automation_requests	Automation Requests
availability_zones	Availability Zones
categories	Categories
chargebacks	Chargebacks
cloud_networks	Cloud Networks
cloud_object_store_containers	Cloud Object Store Containers
cloud_subnets	Cloud Subnets
cloud_templates	Cloud Templates
cloud_tenants	Cloud Tenants
cloud_volume_types	Cloud Volume Types
cloud_volumes	Cloud Volumes
clusters	Clusters
conditions	Conditions
configuration_profiles	Configuration Profiles
configuration_script_payloads	Configuration Script Payloads
configuration_script_sources	Configuration Script Source
configuration_scripts	Configuration Scripts
configured_systems	Configured Systems
container_groups	Container Groups
container_images	Container Images
container_nodes	Container Nodes
container_projects	Container Projects
container_templates	Container Templates
container_volumes	Container Volumes
containers	Containers
conversion_hosts	Conversion Hosts
currencies	Currencies
custom_button_sets	Custom Button Sets
custom_buttons	Custom Buttons
customization_scripts	Customization Scripts
customization_templates	Customization Templates
data_stores	Datastores
enterprises	Enterprises
event_streams	Event Streams
events	Events
features	Product Features
firmware_registries	Firmware Registries
firmwares	Firmwares
flavors	Flavors
floating_ips	Floating IPs
generic_object_definitions	Generic Object Definitions
generic_objects	Generic Objects
groups	Groups

Collection	Description
guest_devices	Guest Devices
hosts	Hosts
instances	Instances
lans	Lans
load_balancers	Load Balancers
measures	Measures
metric_rollups	Metric Rollups
network_routers	Network Routers
notifications	User’s past notifications
orchestration_stacks	Orchestration Stacks
orchestration_templates	Orchestration Template
physical_chassis	Physical Chassis
physical_racks	Physical Racks
physical_servers	Physical Servers
physical_storages	Physical Storages
physical_switches	Physical Switches
pictures	Pictures
policies	Policies
policy_actions	Actions
policy_profiles	Policy Profiles
providers	Providers
provision_dialogs	Provisioning Dialogs
provision_requests	Provision Requests
pxe_images	PXE Images
pxe_menus	PXE Menus
pxe_servers	PXE Servers
rates	Chargeback Rates
regions	Regions
reports	Reports
request_tasks	Request Tasks
requests	Requests
resource_pools	Resource Pools
results	Report Results
roles	Roles
search_filters	Search Filters
security_groups	Security Groups
servers	EVM Servers
service_catalogs	Service Catalogs
service_dialogs	Service Dialogs
service_offerings	Service Offerings
service_orders	Service Orders
service_parameters_sets	Service Parameters Sets
service_requests	Service Requests
service_templates	Service Templates
services	Services
settings	Settings
switches	Switches
tags	Tags
tasks	Tasks
templates	Templates
tenant_groups	Tenant Groups
tenants	Tenants
transformation_mappings	Transformation Mappings
users	Users
vms	Virtual Machines
widgets	Miq Widgets
zones	Zones

- Actions
  - Querying Actions
  - Creating Actions
  - Editing Actions
  - Deleting Actions

Actions

Management of actions is provided via the following collection:

```
/api/actions
```

Full CRUD actions on resources is available:

- Querying Actions
- Creating Actions
- Editing Actions
- Deleting Actions

Querying Actions

Querying all actions in the system is simply:

```
GET /api/actions
```

Getting details on a specific Action:

```
GET /api/actions/:id
```

Creating Actions

Actions can be created via a POST to the actions collection or via the **create** action signature which also allows creation of multiple actions in a single request.

```
POST /api/actions
```

```
{
  "name" : "sample_action",
  "description" : "Sample Action",
  "action_type" : "custom_automation",
  "options" : {
    "ae_message" : "message",
    "ae_request" : "request",
    "ae_hash" : { "sample_key" : "sample_value" }
  }
}
```

or creating multiple actions:

```
{
  "action" : "create",
  "resources" : [
    { "name" : "action_1", "description" : "Action 1", ... },
    { "name" : "action_2", "description" : "Action 2", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating actions.

Editing Actions

```
POST /api/actions/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "Updated Description"
  }
}
```

or editing multiple actions:

```
POST /api/actions
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/actions/101",
      "description" : "Updated Description 1"
    },
    {
      "href" : "http://localhost:3000/api/actions/102",
      "description" : "Updated Description 2"
    },
    ...
  ]
}
```

Deleting Actions

Actions can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/actions/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/actions/101
```

Deleting multiple actions can be done as follows:

```
POST /api/actions
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/actions/101" },
    { "href" : "http://localhost:3000/api/actions/102" },
    ...
  ]
}
```


- Alert Management
  - Querying Alerts
  - Querying Alert Actions
  - Creating Alert Actions
  - Querying Alert Definitions
  - Creating Alert Definitions
  - Editing Alert Definitions
  - Deleting Alert Definitions

Alert Management

Management of alerts, alert actions and alert definitions is provided via the following collections and subcollections:

```
/api/alerts
/api/alerts/:id/alert_actions

/api/alert_definitions
```

Queries of alerts:

- Querying Alerts

Queries and creation of alert actions:

- Querying Alert Actions
- Creating Alert Actions

Full CRUD actions on alert definitions:

- Querying Alert Definitions
- Creating Alert Definitions

- Editing Alert Definitions
- Deleting Alert Definitions

Querying Alerts

Querying all alerts in the system is simply:

```
GET /api/alerts
```

Getting details on a specific alert:

```
GET /api/alerts/:id
```

Querying Alert Actions

Querying all alert actions of a specific alert is done via the subcollection as follows:

```
GET /api/alerts/:alert_id/alert_actions
```

Querying an alert action of a specific alert is done via the subcollection resource as follows:

```
GET /api/alerts/:alert_id/alert_actions/:alert_action_id
```

Creating Alert Actions

Creating an alert action for a specific alert can be done by the following POST:

```
POST /api/alerts/:alert_id/alert_actions
```

```
{
  "action_type" : "custom_automation",
  "comment" : "Comment about the Alert Action"
}
```

Querying Alert Definitions

Querying all alert definitions in the system is simply:

```
GET /api/alert_definitions
```

Getting details on a specific alert definition:

```
GET /api/alert_definitions/:id
```

Creating Alert Definitions

Alert definitions can be created via a POST to the alert definitions collection or via the **create** action signature which also allows creation of multiple alert definitions in a single request.

```
POST /api/alert_definitions
```

```
{
  "description" : "Test Alert Definition",
  "db" : "ContainerNode",
  "expression" : { "eval_method" : "dwh_generic", "mode" : "internal", "options" : {} },
  "options"    : { "notifications" : { "delay_next_evaluation" : 600, "evm_event" : {} } },
  "enabled"    : true
}
```

or creating multiple alert definitions:

```
{
  "action" : "create",
  "resources" : [
    { "description" : "Sample Alert Definition 1", ... },
    { "description" : "Sample Alert Definition 2", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating alert definitions.

Editing Alert Definitions

```
POST /api/alert_definitions/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "options" : { "notifications" : { "delay_next_evaluation" : 1200, "evm:event" : {} } }
  }
}
```

or editing multiple alert definitions:

```
POST /api/alert_definitions
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/alert_definitions/101",
      "options" : { "notifications" : { "delay_next_evaluation" : 1200, "evm:event" : {} } }
    },
    {
      "href" : "http://localhost:3000/api/alert_definitions/102",
      "options" : { "notifications" : { "delay_next_evaluation" : 1200, "evm:event" : {} } }
    },
    ...
  ]
}
```

Deleting Alert Definitions

Alert definitions can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/alert_definitions/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/alert_definitions/101
```

Deleting multiple alert definitions can be done as follows:

```
POST /api/alert_definitions
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/alert_definitions/101" },
    { "href" : "http://localhost:3000/api/alert_definitions/102" },
    ...
  ]
}
```


- Authentications
  - Querying Authentications
  - Creating Authentications
  - Editing Authentications
  - Refreshing Authentications
  - Deleting Authentications

Authentications

Management of authentications is provided via the following collection:

```
/api/authentications
```

Full CRUD actions on authentications is available:

- Querying Authentications
- Creating Authentications
- Editing Authentications
- Refreshing Authentications
- Deleting Authentications

Querying Authentications

Querying all authentications in the system is simply:

```
GET /api/authentications
```

Getting details on a specific authentication:

```
GET /api/authentications/:id
```

One can also query authentications of a configuration script payload's manager as follows:

```
GET /api/configuration_script_payloads/:id/authentications
```

Or getting details on a specific authentication:

```
GET /api/configuration_script_payloads/:id/authentications/:authentication_id
```

Creating Authentications

Authentications can be created via a POST to the authentications collection or via the **create** action signature which also allows creation of multiple authentications in a single request.

```
POST /api/authentications
```

```
{
  "description" : "Authentication Description",
  "name" : "SomeCredentials",
  "related" : {},
  "type" : "ManageIQ::Providers::AnsibleTower::AutomationManager::Credential",
  "manager_resource" : { "href" : "http://localhost:3000/api/providers/7" }
}
```

or creating multiple authentications:

```
{
  "action" : "create",
  "resources" : [
    { "description" : "System Credentials", "name" : "SystemCreds", ... },
    { "description" : "Admin Credentials",  "name" : "AdminCreds", ... },
    ...
  ]
}
```

Optionally, authentications can be created for a provider of a specific configuration script payload, thus eliminating the need of specifying a provider.

```
POST /api/configuration_script_payloads/:id/authentications
```

```
{
  "description" : "Authentication Description",
  "name" : "SomeCredentials",
  "related" : {},
  "type" : "ManageIQ::Providers::AnsibleTower::AutomationManager::Credential"
}
```

Also supporting bulk creates:

```
{
  "action" : "create",
  "resources" : [
    { "description" : "System Credentials", "name" : "SystemCreds", ... },
    { "description" : "Admin Credentials",  "name" : "AdminCreds", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating authentications.

Editing Authentications

```
POST /api/authentications/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "name" : "UpdatedCredentials"
  }
}
```

or editing multiple authentications:

```
POST /api/authentications
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/authentications/101",
      "description" : "Updated Sample Credentials 1",
      "name" : "UpdatedCredentials1"
    },
    {
      "href" : "http://localhost:3000/api/authentications/102",
      "description" : "Updated Sample Credentials 2",
      "name" : "UpdatedCredentials2"
    },
    ...
  ]
}
```

Refreshing Authentications

Authentications can be refreshed by posting a **refresh** action to a single resource or to multiple resources in bulk by targeting the collection.

```
POST /api/authentications/:id
```

```
{
  "action" : "refresh"
}
```

or refreshing multiple authentications:

```
POST /api/authentications
```

```
{
  "action" : "refresh",
  "resources" : [
    { "id" : "51" },
    { "id" : "52" }
  ]
}
```

Example:

```
POST /api/authentications/51
```

```
{
  "action" : "refresh"
}
```

Response:

```
{
  "success" : true,
  "message" : "Refreshing Authentication id:51 name:'SampleCredentials'",
  "task_id" : "8",
  "task_href" : "http://localhost:3000/api/tasks/8",
  "tasks" : [
    {
      "id" : "8",
      "href" : "http://localhost:3000/api/tasks/8"
    }
  ]
}
```

Deleting Authentications

Authentications can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/authentications/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/authentications/101
```

Deleting multiple authentications can be done as follows:

```
POST /api/authentications
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/authentications/101" },
    { "href" : "http://localhost:3000/api/authentications/102" },
    ...
  ]
}
```


- Automate
  - Usage
  - Querying Domains
  - Querying By Fully Qualified Name
  - Querying First Level Children Nodes
  - Querying Subtree
  - Querying State Machines Subtree

Automate

Querying the Automate model is provided via the following entrypoint:

```
/api/automate
```


- Usage
- Querying Domains
- Querying By Fully Qualified Name
- Querying First Level Children Nodes
- Querying Subtree
- Querying State Machines Subtree

Usage

Querying the Automate Model via /api/automate allows queries of:

- top level Automate Domains
- any object in the Automate Model
- arbitrary depth via the **depth** parameter

depth value	data returned
0	Just node itself
1	Node itself plus its first level children nodes
#	Node itself plus # levels of children nodes
-1	Node itself plus the whole subtree underneath it

- Automate model object tree to include state machine endpoints via the **search_options** parameter with value **state_machines**

Example queries:

```
GET /api/automate
GET /api/automate/manageiq
GET /api/automate/manageiq?depth=2
GET /api/automate/manageiq?depth=-1&search_options=state_machines
```

Querying Domains

Querying the Automate Domains can be done via GET on the /api/automate entrypoint as follows:

```
GET /api/automate
```

```
{
  "name": "automate",
  "subcount": 4,
  "resources": [
    {
      "fqname": "/ManageIQ",
      "domain_fqname": "/",
```

```
      "klass": "MiqAeDomain",
      "id": "2352",
      "name": "ManageIQ",
      "created_on": "2016-10-31T14:58:09Z",
      "updated_on": "2016-10-31T14:58:09Z",
      "description": "Base domain",
      "updated_by": "system",
      "priority": 0,
      "enabled": true,
      "tenant_id": "1",
      "source": "system"
    },
    ...
  ]
}
```

Note:

The fqname is always returned by this query

If desired a sub-set of attributes can be asked via as follows:

```
GET /api/automate?attributes=name,source
```

```
{
  "name": "automate",
  "subcount": 4,
  "resources": [
    {
      "fqname": "/ManageIQ",
      "name": "ManageIQ",
      "source" : "system"
    },
    ...
  ]
}
```

Querying By Fully Qualified Name

Querying any object in the Automate Model by its fully qualified name as follows:

Note:

The fqname in the query is *not* case sensitive

```
GET /api/automate/manageiq
```

```
{
  "name": "automate",
  "subcount": 1,
  "resources": [
    {
      "fqname": "/ManageIQ",
      "domain_fqname": "/",
      "klass": "MiqAeDomain",
      "id": "2352",
      "name": "ManageIQ",
      "created_on": "2016-10-31T14:58:09Z",
      "updated_on": "2016-10-31T14:58:09Z",
      "description": "Base domain",
      "updated_by": "system",
      "priority": 0,
      "enabled": true,
      "tenant_id": "1",
      "source": "system"
    }
  ]
}
```

Another example showing fetching an automate class object:

```
GET /api/automate/manageiq/system/request
```

```
{
  "name": "automate",
  "subcount": 1,
  "resources": [
    {
      "fqname": "/ManageIQ/System/Request",
      "domain_fqname": "/System/Request",
      "klass": "MiqAeClass",
```

```
      "id": "75",
      "description": "Automation Requests",
      "name": "Request",
      "created_on": "2015-12-09T20:56:44Z",
      "updated_on": "2015-12-09T20:56:44Z",
      "namespace_id": "41",
      "updated_by": "system"
    }
  ]
}
```

Automate model nodes returned include both:

- **fqname**: the fully qualified name of the node which is rooted at /.
    - i.e. /ManageIQ/System/Request
- **domain_fqname**: the fully qualified name of the node rooted at the domain, which is **fqname** without the Domain prefix
    - i.e. /System/Request

Querying First Level Children Nodes

When specifying a depth of 1, the Automate model node and its first level children nodes are returned.

In this example, we are asking for the /ManageIQ Domain and its direct children nodes.

```
GET /api/automate/manageiq?depth=1&attributes=domain_fqname,klass
```

```
{
  "name": "automate",
  "subcount": 9,
  "resources": [
    {
      "fqname": "/ManageIQ",
      "domain_fqname": "/",
      "klass": "MiqAeDomain"
    },
    {
      "fqname": "/ManageIQ/System",
      "domain_fqname": "/System",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/Service",
      "domain_fqname": "/Service",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/Infrastructure",
      "domain_fqname": "/Infrastructure",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/GenericObject",
      "domain_fqname": "/GenericObject",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/Deployment",
      "domain_fqname": "/Deployment",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/Control",
      "domain_fqname": "/Control",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/ConfigurationManagement",
      "domain_fqname": "/ConfigurationManagement",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/Cloud",
      "domain_fqname": "/Cloud",
      "klass": "MiqAeNamespace"
    }
  ]
}
```

Querying Subtree

When specifying a depth of -1, the Automate model node and all of the nodes underneath it are returned.

```
GET /api/automate/manageiq?depth=-1&attributes=klass
```

```
{
  "name": "automate",
  "subcount": 935,
  "resources": [
    {
      "fqname": "/ManageIQ",
      "klass": "MiqAeDomain"
    },
    {
      "fqname": "/ManageIQ/System",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/System/Event",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/System/Event/RequestEvent",
      "klass": "MiqAeNamespace"
    },
    ...
  ]
}
```

Querying State Machines Subtree

In this example, we are querying the ManageIQ Automate Domain model tree but only asking for branches that include state machine nodes.

```
GET /api/automate/manageiq?depth=-1&attributes=domain_fqname,klass&search_options=state_machines
```

```
{
  "name": "automate",
  "subcount": 104,
  "resources": [
    {
      "fqname": "/ManageIQ",
      "domain_fqname": "/",
      "klass": "MiqAeDomain"
    },
    {
      "fqname": "/ManageIQ/System",
      "domain_fqname": "/System",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/System/CommonMethods",
      "domain_fqname": "/System/CommonMethods",
      "klass": "MiqAeNamespace"
    },
    {
      "fqname": "/ManageIQ/System/CommonMethods/QuotaStateMachine",
      "domain_fqname": "/System/CommonMethods/QuotaStateMachine",
      "klass": "MiqAeClass"
    },
    {
      "fqname": "/ManageIQ/System/CommonMethods/QuotaStateMachine/quota",
      "domain_fqname": "/System/CommonMethods/QuotaStateMachine/quota",
      "klass": "MiqAeInstance"
    },
    ...
  ]
}
```


- Automate Domains
  - Querying Domains
  - Accessing Domains
  - Refresh From Source

Automate Domains

Querying the Automate Domains and triggering actions on domains is provided via the following entrypoint:

```
/api/automate_domains
```

- Querying Domains
- Accessing Domains
- Refresh From Source

Querying Domains

Querying the Automate Domains can be done via GET on the /api/automate_domains entrypoint and can be queried like any other collection:

```
GET /api/automate_domains
```

```
{
  "name": "automate_domains",
  "count": 9,
  "subcount": 9,
  "resources": [
    {
      "href": "http://localhost:3000/api/automate_domains/2352"
    },
    {
      "href": "http://localhost:3000/api/automate_domains/2347"
    },
    ...
  ],
  "actions": [
    {
      "name": "query",
      "method": "post",
      "href": "http://localhost:3000/api/automate_domains"
    },
    {
      "name": "refresh_from_source",
      "method": "post",
      "href": "http://localhost:3000/api/automate_domains"
    }
  ]
}
```

Accessing Domains

Accessing individual domains can be done via their numeric id or their name (*case sensitive*) as follows:

```
GET /api/automate_domains/2352
```

```
GET /api/automate_domains/ManageIQ
```

Refresh From Source

When Automate domains are imported from external sources, i.e. Git, the *refresh_from_source* will re-fetch the Automate domain from external source and refresh the automate domain. This will refetch the Domain from the same reference, i.e. branch or tag and name.

```
POST /api/automate_domains/SampleDomain
```

```
{
  "action": "refresh_from_source"
}
```

All refresh_from_source actions are done asynchronously and the related Task id and href are returned as follows:

```
{
  "success": true,
  "message": "Refreshing Automate Domain id:2347 name:'SampleDomain' from git repository",
  "task_id": "61",
  "task_href": "http://localhost:3000/api/tasks/61",
  "href": "http://localhost:3000/api/automate_domains/SampleDomain"
}
```

If the Domain needs to be refreshed from a different branch or tag and name, the following can be passed in with the *refresh_from_source* action:

```
{
  "action": "refresh_from_source",
  "resource" : {
    "ref_type" : "branch" | "tag",
    "ref" : "other_branch_or_tag_name"
  }
}
```

- Automation Requests
  - Creating Requests

- Editing Requests
- Approving or Denying Requests
  - Approving a Request
  - Denying a Request
- Monitoring Requests
  - Response:
- Editing Request Tasks

Automation Requests

Automation requests are made available via the following entrypoint:

`/api/automation_requests`

- Creating Requests
- Editing Requests
- Approving or Denying Requests
- Monitoring Requests
- Editing Request Tasks

Creating Requests

Either by specifying a *create* action or by posting the automation request directly to **/api/automation_requests**.

As an example:

```
{
  "uri_parts" : {
    "namespace" : "System",
    "class"     : "Request",
    "instance"  : "InspectME",
    "message"   : "create"
  },
  "parameters" : {
    "var1" : "value 1",
    "var2" : "value 2",
    "minimum_memory" : 2048
  },
  "requester" : {
    "auto_approve" : true
  }
}
```

Editing Requests

Updating requests can be performed by posting an *edit* action as follows:

```
POST /api/automation_requests/11
```

```
{
  "action" : "edit",
  "resource" : {
    "options" : {
      "attrs" : {
        "var1" : "updated value1",
        "var2" : "updated value2",
        "minimum_memory" : 4096
      }
    }
  }
}
```

Response:

```
{
  "href": "http://localhost:3000/api/automation_requests/11",
  "id": "11",
  "options": {
    "message": "create",
    "namespace": "System",
    "class_name": "Request",
    "instance_name": "InspectME",
    "user_id": "1",
```

```
    "attrs": {
      "var1" : "updated value1",
      "var2" : "updated value2",
      "minimum_memory" : 4096
    }
  },
  "tenant_id": "1",
  "approval_state": "approved",
  "status": "Ok",
  "request_type": "automation",
  "description": "Automation Task",
  "type": "AutomationRequest",
  "created_on": "2018-01-08T13:48:00Z",
  "updated_on": "2018-01-08T13:56:15Z",
  "requester_id": "1",
  "requester_name": "Administrator",
  "request_state": "pending",
  "message": "Automation Request - Request Created",
  "userid": "admin",
  "process": true
}
```

Approving or Denying Requests

If an automation request is not auto approved, it can be approved or denied via the actions *approve* and *deny* on the automation request resource.

Approving a Request

```
POST /api/automation_requests/:id

{
  "action" : "approve",
  "reason" : "reason for approval"
}
```

Denying a Request

```
POST /api/automation_requests/:id

{
  "action" : "deny",
  "reason" : "reason for denial"
}
```

Monitoring Requests

Once an automation request is created, the response will be the queryable automation request itself, i.e. **/api/automation_request/:id**

Response:

```
{
  "results": [
    {
      "id": "10",
      "description": "Automation Task",
      "approval_state": "approved",
      "type": "AutomationRequest",
      "created_on": "2015-04-14T18:12:12Z",
      "updated_on": "2015-04-14T18:12:12Z",
      "requester_id": "1",
      "requester_name": "Administrator",
      "request_type": "automation",
      "request_state": "pending",
      "status": "Ok",
      "options": {
        "message": "create",
        "namespace": "System",
        "class_name": "Request",
        "instance_name": "InspectME",
        "user_id": "1",
        "attrs": {
          "var1": "value 1",
          "var2": "value 2",
          "minimum_memory": 2048,
          "userid": "admin"
        }
      },
      "userid": "admin"
    }
```

```
  ]
}
```

In the above example, the request could be queried periodically until the **request_state** reaches the **finished** state.

Note:

The request tasks of an automation request can also be queried by expanding the request_tasks subcollection as follows:

```
GET /api/automation_requests/:id?expand=request_tasks
```

or by fetching them directly as follows:

```
GET /api/automation_requests/:id/request_tasks?expand=resources
```

Editing Request Tasks

Request tasks for Automation Requests can be updated as follows:

```
POST /api/automation_requests/:id/request_tasks/:request_task_id
```

```
{
  "action" : "edit",
  "resource" : {
    "options" : {
      "request_param_a" : "value_a",
      "request_param_b" : "value_b"
    }
  }
}
```


- Automate Workspaces
  - Querying Automate Workspaces
    - Request:
    - Response:
  - Editing Automate Workspaces
  - Encrypting Automate Workspace Passwords
    - Request:
    - Response:
  - Decrypting Automate Workspace Passwords
    - Request:
    - Response:

Automate Workspaces

Management of Automate Workspaces is provided with the following collection:

```
/api/automate_workspaces
```


- Querying Automate Workspaces
- Editing Automate Workspaces
- Encrypting Automate Workspace Passwords
- Decrypting Automate Workspace Passwords

Querying Automate Workspaces

It is not allowed to query the automate workspace collection, only individual automate workspaces can be fetched by guid.

```
GET /api/automate_workspaces/:guid
```

Example GET Request and Response:

Request:

```
GET /api/automate_workspaces/430a66a0-b564-4341-b9b2-85d6f467f26f
```

Response:

```
{
  "href": "http://localhost:3000/api/automate_workspaces/430a66a0-b564-4341-b9b2-85d6f467f26f",
  "id": "2",
  "guid": "430a66a0-b564-4341-b9b2-85d6f467f26f",
```

```
  "user_id": "1",
  "tenant_id": "1",
  "input": {
    "objects": {
      "root": {
        "var1": "a",
        "var2": "b"
      }
    },
    "method_parameters": {
      "arg1": "first_argument"
    }
  },
  "actions": [
    {
      "name": "edit",
      "method": "post",
      "href": "http://localhost:3000/api/automate_workspaces/430a66a0-b564-4341-b9b2-85d6f467f26f"
    },
    {
      "name": "encrypt",
      "method": "post",
      "href": "http://localhost:3000/api/automate_workspaces/430a66a0-b564-4341-b9b2-85d6f467f26f"
    },
    {
      "name": "decrypt",
      "method": "post",
      "href": "http://localhost:3000/api/automate_workspaces/430a66a0-b564-4341-b9b2-85d6f467f26f"
    }
  ]
}
```

Editing Automate Workspaces

Updating Automate Workspaces can be done via the *edit* action as follows:

```
POST /api/automate_workspaces/:guid
```

```
{
  "action" : "edit",
  "resource" : {
    "objects" : {
      "root" : {
        "var3" : "c"
      }
    }
  }
}
```

Encrypting Automate Workspace Passwords

Encrypting passwords in the automate workspace can be done as follows:

Request:

```
POST /api/automate_workspaces/:guid
```

```
{
  "action" : "encrypt",
  "resource" : {
    "object" : "method_parameters",
    "attribute" : "arg2",
    "value" : "unencrypted_password"
  }
}
```

Response:

```
{
  "href": "http://localhost:3000/api/automate_workspaces/430a66a0-b564-4341-b9b2-85d6f467f26f",
  "id": "2",
  "guid": "430a66a0-b564-4341-b9b2-85d6f467f26f",
  "user_id": "1",
  "tenant_id": "1",
  "input": {
    "objects": {
      "root": {
        "var1": "a",
        "var2": "b"
      }
```

```
    },
    "method_parameters": {
      "arg1": "first_argument",
      "arg2": "password::********"
    }
  }
}
```

Decrypting Automate Workspace Passwords

Encrypted passwords in the automate workspace can be decrypted individually as follows:

Request:

```
POST /api/automate_workspaces/:guid

{
  "action" : "decrypt",
  "resource" : {
    "object" : "method_parameters",
    "attribute" : "arg2"
  }
}
```

Response:

```
{
  "object" : "method_parameters",
  "attribtue" : "arg2",
  "value" : "unencrypted_password"
}
```


- Categories & Tags
  - Querying Categories
  - Querying Tags
    - Request:
    - Response:
    - Request:
    - Response:
  - Creating Categories
    - Response:
  - Creating Tags
  - Editing Categories & Tags
  - Deleting Categories & Tags
    - Request:
    - Response:

Categories & Tags

Management of Categories and Tags is provided via the following collections:

`/api/categories`

`/api/tags`

Full CRUD actions on both cateogries and tags are made available.

- Querying Categories
- Querying Tags
- Creating Categories
- Creating Tags
- Editing Categories & Tags
- Deleting Categories & Tags

Querying Categories

Querying all categories in the system is simply:

```
GET /api/categories
```

Or getting attributes of interest like *name*, *description* or *ns* for the namespace:

```
GET /api/categories?expand=resources&attributes=ns,name,description
```

Querying Tags

Querying all tags in the system is simply:

```
GET /api/tags
```

Querying tags of a specific category is done as follows:

```
GET /api/categories/:id/tags
```

Example Query for Location Category:

Request:

```
GET /api/categories/1
```

Response:


```
{
  "href": "http://localhost:3000/api/categories/1",
  "id": "1",
  "description": "Location",
  "read_only": false,
  "syntax": "string",
  "single_value": true,
  "example_text": "The geographic location of the resource, such as New York, Chicago, or London.",
  "tag_id": "4",
  "parent_id": "0",
  "show": true,
  "default": true,
  "name": "location",
  "actions": [
    {
      "name": "edit",
      "method": "post",
      "href": "http://localhost:3000/api/categories/1"
    },
    {
      "name": "delete",
      "method": "post",
      "href": "http://localhost:3000/api/categories/1"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "http://localhost:3000/api/categories/1"
    }
  ]
}
```


Querying all tags of the Location category:

Request:

```
GET /api/categories/1/tags?expand=resources&attributes=name
```

Response:


```
{
  "name": "tags",
  "count": 1389,
  "subcount": 4,
  "resources": [
    {
      "href": "http://localhost:3000/api/categories/1/tags/8",
      "id": "8",
      "name": "/managed/location/paris"
    },
    {
      "href": "http://localhost:3000/api/categories/1/tags/7",
      "id": "7",
      "name": "/managed/location/london"
    },
    {
```

```
      "href": "http://localhost:3000/api/categories/1/tags/6",
      "id": "6",
      "name": "/managed/location/chicago"
    },
    {
      "href": "http://localhost:3000/api/categories/1/tags/5",
      "id": "5",
      "name": "/managed/location/ny"
    }
  ],
  "actions": [
    {
      "name": "create",
      "method": "post",
      "href": "http://localhost:3000/api/categories/1/tags"
    },
    {
      "name": "delete",
      "method": "post",
      "href": "http://localhost:3000/api/categories/1/tags"
    }
  ]
}
```

For additional details on tags, the **category** and **classification** attributes can be queried on tags as follows:

```
GET /api/tags/5?attributes=category,classification
```

Additionally, a compound attribute called **categorization** is provided to return key information from both **category** and **classification** of tags as follows:

```
GET /api/tags/5?attributes=categorization
```

```
{
  "href": "http://localhost:3000/api/tags/5",
  "id": "5",
  "name": "/managed/location/ny",
  "categorization": {
    "name": "ny",
    "description": "New York",
    "category": {
      "name": "location",
      "description": "Location"
    },
    "display_name": "Location: New York"
  }
}
```

Creating Categories

Creating Categories can be done by posting the Category JSON to the categories collection directly, or via the **create** action signature as follows:

```
POST /api/categories
```

```
{
  "name" : "test",
  "description" : "Test Category"
}
```

or via the **create** action:

```
{
  "action" : "create",
  "resource" : {
    "name" : "test",
    "description" : "Test Category"
  }
}
```

Response:

```
{
  "results": [
    {
      "id": "1430",
      "description": "Test Category",
      "read_only": false,
      "syntax": "string",
      "single_value": false,
      "tag_id": "1439",
      "parent_id": "0",
```

```
      "show": true
    }
  ]
}
```

Attributes which can also be specified when creating categories include:

example_text
show
single_value
syntax

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Categories.

Creating Tags

New tags for a category can be created either as a tag collection post or via a tag subcollection **create** action to a specific category.

```
POST /api/tags
```

```
{
  "name" : "test_tag",
  "description" : "Test Tag",
  "category" : { "href" : "http://localhost:3000/api/categories/1430" }
}
```

The **category** above can identify a category by specifying either one of the **href**, **id** or **name** attributes.

Tags can optionally be created via the **create** action on the tags subcollection of a category as follows:

```
POST /api/categories/1430/tags
```

```
{
  "action" : "create",
  "resource" : {
    "name" : "test_tag",
    "description" : "Test Tag"
  }
}
```

or simply:

```
POST /api/categories/1430/tags
```

```
{
  "name" : "test_tag",
  "description" : "Test Tag"
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Tags.

Editing Categories & Tags

Editing Categories and Tags can be done via the **edit** action:

Example editing a category:

```
POST /api/categories/1430
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "Updated Category Description"
  }
}
```

Example editing a tag:

```
POST /api/tags/1441
```

```
{
  "action" : "edit",
  "resource" : {
    "name" : "updated_test_tag"
```

```
  }
}
```

Deleting Categories & Tags

Deleting Categories and Tags can be done via either the **delete** post action or the DELETE HTTP method.

```
POST /api/categories/1430

{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/categories/1430
```

Deleting associated tag example:

```
POST /api/tags/1441

{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/tags/1441
```

One can also delete the tag when accessed via the subcollection

Request:

```
POST /api/categories/1430/tags

{
  "action" : "delete",
  "resources" : [
    { "id" : "1441" },
    { "id" : "1442" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "tags id: 1441 deleting"
    },
    {
      "success": true,
      "message": "tags id: 1442 deleting"
    }
  ]
}
```

Tags to be deleted can be specified via the **href**, **id** or **name** attribute.

- Chargebacks, Rates & Assignments
  - Querying Chargebacks
    - Request:
    - Response:
    - Request:
    - Response:
  - Creating Chargeback Rates
    - Request:
    - Response:
  - Updating Chargeback Rates
    - Request:
    - Response:
  - Deleting Chargeback Rates
    - Request:
    - Response:
  - Querying Assigned Chargeback Rates

        - Assigned chargeback rates on single rate:
        - Response:
        - Assigned chargeback rates on collection:
        - Response:
    - Assign action for Chargeback Rates
    - “Compute” Rate Type
    - “Storage” Rate Type
        - Response:
    - Basic Template for single chargeback request
    - Basic Template for chargebacks collection request
    - Enterprise
    - Selected Providers
    - Selected Clusters
    - Tagged VMs and Instances
    - Tagged Container Images
    - Labeled Container Images
    - Tenants
    - Enterprise
    - Selected Datastores
    - Tagged Datastores
    - Tenants
    - Unassign action for Chargeback Rates
    - Example of Chargeback Flow
    - Create Chargeback Report
        - Chargeback Report Options
    - Filtering according to resource
        - Chargeback for VMs (db=ChargebackVm)
        - Chargeback for Projects (db=ChargebackContainerProject)
        - Chargeback for Container Images (db=ChargebackContainerImage)
    - Grouping Report Options
        - Chargeback for VMs (db=ChargebackVm)
        - Chargeback for Projects (db=ChargebackContainerProject)
        - Chargeback for Container Images (db=ChargebackContainerImage)
        - Example of request to create chargeback report
    - Create Chargeback Rates
    - Create Chargebacks (top-level rate entity)
    - Assign Chargeback Rate
    - Run Chargeback Report
    - Results of Chargeback Report

Chargebacks, Rates & Assignments

Queries of Chargebacks and Management of Rates is provided via the following collections:

`/api/chargebacks`

`/api/rates`

- Querying Chargebacks
- Creating Chargeback Rates
- Updating Chargeback Rates
- Deleting Chargeback Rates
- Querying Assigned Chargeback Rates
- Assign action for Chargeback Rates
- Unassign action for Chargeback Rates
- Example of Chargeback Flow

Querying Chargebacks

Querying all chargebacks in the system is simply:

```
GET /api/chargebacks
```

or Query a specific chargeback, for example:

```
GET /api/chargebacks/1
```


```
{
  "href": "http://localhost:3000/api/chargebacks/1",
  "id": "1",
  "guid": "b47a0ef0-4335-11df-aba8-001d09066d98",
  "description": "Default",
  "rate_type": "Compute",
  "created_on": "2014-11-20T19:10:03Z",
  "updated_on": "2014-11-20T19:10:03Z",
  "default": true
}
```


Optionally, one can query the rates details for a chargeback as follows:

```
GET /api/chargebacks/:id/rates
```

Example getting additional rates details(attributes)

Request:

```
GET /api/chargebacks/1/rates?expand=resources&attributes=description,metric,group,source,friendly_rate
```

Response:


```
{
  "name": "rates",
  "count": 10,
  "subcount": 8,
  "resources": [
    {
      "href": "http://localhost:3000/api/chargebacks/1/rates/8",
      "id": "8",
      "description": "Fixed Compute Cost 2",
      "group": "fixed",
      "source": "compute_2",
      "friendly_rate": "0.0 Hourly"
    },
    {
      "href": "http://localhost:3000/api/chargebacks/1/rates/7",
      "id": "7",
      "description": "Fixed Compute Cost 1",
      "group": "fixed",
      "source": "compute_1",
      "friendly_rate": "1.0 Hourly"
    },
    {
      "href": "http://localhost:3000/api/chargebacks/1/rates/6",
      "id": "6",
      "description": "Used Disk I/O in KBps",
      "group": "disk_io",
      "metric": "disk_usage_rate_average",
      "source": "used",
      "friendly_rate": "Hourly @ 0.0 + 1.0 per Kbps from 0.0 to Infinity"
    },
    {
      "href": "http://localhost:3000/api/chargebacks/1/rates/5",
      "id": "5",
      "description": "Used Network I/O in KBps",
      "group": "net_io",
      "metric": "net_usage_rate_average",
      "source": "used",
      "friendly_rate": "Hourly @ 0.0 + 0.0 per Kbps from 0.0 to Infinity"
    },
    {
      "href": "http://localhost:3000/api/chargebacks/1/rates/4",
      "id": "4",
      "description": "Allocated Memory in MB",
      "group": "memory",
      "metric": "derived_memory_available",
      "source": "allocated",
      "friendly_rate": "Hourly @ 0.0 + 1.0 per Megabytes from 0.0 to Infinity"
    },
    {
      "href": "http://localhost:3000/api/chargebacks/1/rates/3",
      "id": "3",
```

```
      "description": "Used Memory in MB",
      "group": "memory",
      "metric": "derived_memory_used",
      "source": "used",
      "friendly_rate": "Hourly @ 0.0 + 0.0 per Megabytes from 0.0 to Infinity"
    },
    {
      "href": "http://localhost:3000/api/chargebacks/1/rates/2",
      "id": "2",
      "description": "Allocated CPU Count",
      "group": "cpu",
      "metric": "derived_vm_numvcpus",
      "source": "allocated",
      "friendly_rate": "Hourly @ 0.0 + 1.0 per Cpu from 0.0 to Infinity"
    },
    {
      "href": "http://localhost:3000/api/chargebacks/1/rates/1",
      "id": "1",
      "description": "Used CPU in MHz",
      "group": "cpu",
      "metric": "cpu_usagemhz_rate_average",
      "source": "used",
      "friendly_rate": "Hourly @ 0.0 + 0.0 per Megahertz from 0.0 to Infinity"
    }
  ]
}
```

Example getting additional of rates details(relations)

Request:

```
GET /api/rates?
expand=resources&attributes=chargeback_rate,detail_measure,detail_currency,chargeable_field,chargeback_tiers
```

Response:

```
{
    "name": "rates",
    "count": 138,
    "subcount": 138,
    "pages": 1,
    "resources": [
        {
            "href": "http://localhost:3090/api/rates/544",
            "id": "544",
            "enabled": true,
            "description": "Allocated CPU Count",
            "group": "cpu",
            "source": null,
            "metric": "derived_vm_numvcpus",
            "per_time": "hourly",
            "per_unit": "cpu",
            "friendly_rate": "Hourly @ 1.0 + 0.0 per Cpu from 0.0 to Infinity",
            "chargeback_rate_id": "8",
            "created_on": "2018-06-01T11:24:52Z",
            "updated_on": "2018-06-01T11:24:52Z",
            "chargeback_rate_detail_measure_id": null,
            "chargeback_rate_detail_currency_id": "1",
            "chargeable_field_id": "3",
            "sub_metric": null,
            "chargeback_rate": {
                "id": "8",
                "guid": "52a70f67-7ab8-47a0-ba36-7808ac2449e0",
                "description": "solinux1",
                "rate_type": "Compute",
                "created_on": "2018-06-01T11:24:52Z",
                "updated_on": "2018-06-01T11:24:52Z",
                "default": false
            },
            "detail_currency": {
                "id": "1",
                "code": "USD",
                "name": "United States Dollar",
                "full_name": "United States Dollar",
                "symbol": "$",
                "unicode_hex": "36",
                "created_at": "2019-06-13T18:10:32Z",
                "updated_at": "2019-07-09T13:42:36Z"
            },
            "chargeable_field": {
```

```
                "id": "3",
                "chargeback_rate_detail_measure_id": null,
                "metric": "derived_vm_numvcpus",
                "group": "cpu",
                "source": "allocated",
                "description": "Allocated CPU Count"
            },
            "chargeback_tiers": [
                {
                    "id": "11010",
                    "chargeback_rate_detail_id": "544",
                    "start": 0.0,
                    "finish": null,
                    "fixed_rate": 1.0,
                    "variable_rate": 0.0
                }
            ],
            "actions": [
                {
                    "name": "edit",
                    "method": "post",
                    "href": "http://localhost:3090/api/rates/544"
                },
                {
                    "name": "edit",
                    "method": "patch",
                    "href": "http://localhost:3090/api/rates/544"
                },
                {
                    "name": "edit",
                    "method": "put",
                    "href": "http://localhost:3090/api/rates/544"
                },
                {
                    "name": "delete",
                    "method": "post",
                    "href": "http://localhost:3090/api/rates/544"
                },
                {
                    "name": "delete",
                    "method": "delete",
                    "href": "http://localhost:3090/api/rates/544"
                }
            ]
        },
        {
            "href": "http://localhost:3090/api/rates/510",
            "id": "510",
            "enabled": true,
            "description": "Allocated CPU Count",
            "group": "cpu",
            "source": null,
            "metric": null,
            "per_time": "hourly",
            "per_unit": "cpu",
            "friendly_rate": "Hourly @ 1.0 + 0.0 per Cpu from 0.0 to Infinity",
            "chargeback_rate_id": "7",
            "created_on": "2018-03-28T10:00:13Z",
            "updated_on": "2018-03-28T10:00:13Z",
            "chargeback_rate_detail_measure_id": null,
            "chargeback_rate_detail_currency_id": "3",
            "chargeable_field_id": "3",
            "sub_metric": null,
            "chargeback_rate": {
                "id": "7",
                "guid": "a72fd416-26ec-4aa0-bc94-0b741379dfcd",
                "description": "teamA_compute_all",
                "rate_type": "Compute",
                "created_on": "2018-03-28T10:00:13Z",
                "updated_on": "2018-03-28T10:00:13Z",
                "default": false
            },
            "detail_currency": {
                "id": "3",
                "code": "GBP",
                "name": "British Pound",
                "full_name": "British Pound",
                "symbol": "£",
                "unicode_hex": "163",
                "created_at": "2019-06-13T18:10:32Z",
                "updated_at": "2019-07-09T13:42:35Z"
```

```
            },
            "chargeable_field": {
                "id": "3",
                "chargeback_rate_detail_measure_id": null,
                "metric": "derived_vm_numvcpus",
                "group": "cpu",
                "source": "allocated",
                "description": "Allocated CPU Count"
            },
            "chargeback_tiers": [
                {
                    "id": "10115",
                    "chargeback_rate_detail_id": "510",
                    "start": 0.0,
                    "finish": 'infinity',
                    "fixed_rate": 1.0,
                    "variable_rate": 0.0
                }
            ]
        },
        {
            "href": "http://localhost:3090/api/rates/1859",
            "id": "1859",
            "enabled": true,
            "description": "My CPU allocation rate",
            "group": "cpu",
            "source": "allocated",
            "metric": null,
            "per_time": "daily",
            "per_unit": "megahertz",
            "friendly_rate": "Daily @ 2.0 + 0.0 per Megahertz from 0.0 to 1.0\nDaily @ 2.0 + 0.0 per
Megahertz from 1.0 to Infinity",
            "chargeback_rate_id": "1",
            "created_on": "2020-05-26T10:34:39Z",
            "updated_on": "2020-05-26T10:34:39Z",
            "chargeback_rate_detail_measure_id": "1",
            "chargeback_rate_detail_currency_id": "5",
            "chargeable_field_id": "3",
            "sub_metric": null,
            "chargeback_rate": {
                "id": "1",
                "guid": "b47a0ef0-4335-11df-aba8-001d09066d98",
                "description": "Default",
                "rate_type": "Compute",
                "created_on": "2020-04-23T14:43:50Z",
                "updated_on": "2020-04-23T15:00:29Z",
                "default": true
            },
            "detail_measure": {
                "id": "1",
                "name": "Bytes Units",
                "units": [
                    "bytes",
                    "kilobytes",
                    "megabytes",
                    "gigabytes",
                    "terabytes"
                ],
                "units_display": [
                    "B",
                    "KB",
                    "MB",
                    "GB",
                    "TB"
                ],
                "step": 1024,
                "created_at": "2020-04-23T14:43:50Z",
                "updated_at": "2020-04-23T15:00:28Z"
            },
            "detail_currency": {
                "id": "5",
                "code": "AUD",
                "name": "Australian Dollar",
                "full_name": "Australian Dollar",
                "symbol": "$",
                "unicode_hex": null,
                "created_at": "2019-06-13T18:10:32Z",
                "updated_at": "2019-07-09T13:42:35Z"
            },
            "chargeable_field": {
                "id": "3",
```

```
                "chargeback_rate_detail_measure_id": null,
                "metric": "derived_vm_numvcpus",
                "group": "cpu",
                "source": "allocated",
                "description": "Allocated CPU Count"
            },
            "chargeback_tiers": [
                {
                    "id": "13114",
                    "chargeback_rate_detail_id": "1859",
                    "start": 0.0,
                    "finish": 1.0,
                    "fixed_rate": 2.0,
                    "variable_rate": 0.0
                },
                {
                    "id": "13115",
                    "chargeback_rate_detail_id": "1859",
                    "start": 1.0,
                    "finish": 'infinity',
                    "fixed_rate": 2.0,
                    "variable_rate": 0.0
                }
            ]
        },
        ...
```

Creating Chargeback Rates

Creating a Chargeback Rate is done by posting a new resource or *create* action to the rates collection.

Example follows:

Request:

```
POST /api/rates
```

```
{
  "per_time" : "daily",
  "chargeback_rate_id" : "1",
  "chargeable_field_id" : "1",
  "description": "My CPU allocation rate",
  "group" : "cpu",
  "per_unit" : "megahertz",
  "source" : "allocated",
  "chargeback_tiers": [
       {
          "start": 0.0,
          "finish": "infinity",
          "fixed_rate": 2.0,
          "variable_rate": 0.0
        }
   ]
}
```

Response:

```
{
    "results": [
        {
            "href": "http://localhost:3090/api/rates/1866",
            "id": "1866",
            "enabled": true,
            "description": "My CPU allocation rate",
            "group": "cpu",
            "source": "allocated",
            "metric": null,
            "per_time": "daily",
            "per_unit": "megahertz",
            "friendly_rate": "Daily @ 2.0 + 0.0 per MHz from 0.0 to Infinity",
            "chargeback_rate_id": "1",
            "created_on": "2020-05-28T13:28:15Z",
            "updated_on": "2020-05-28T13:28:15Z",
            "chargeback_rate_detail_measure_id": null,
            "chargeback_rate_detail_currency_id": null,
            "chargeable_field_id": "1",
            "sub_metric": null
        }
    ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Chargeback Rates.

Updating Chargeback Rates

Updating rates can be done by posting **edit** actions on the rates resource.

Request:

```
POST /api/rates/16

{
  "action" : "edit",
  "resource" : { "description" : "Allocated NICs", "per_time": "hourly" }
}
```

Response:

```
{
    "href": "http://localhost:3090/api/rates/16",
    "id": "16",
    "description": "Allocated NICs",
    "per_time": "hourly",
    "chargeback_rate_id": "1",
    "chargeable_field_id": "3",
    "enabled": true,
    "group": "net_io",
    "source": null,
    "metric": null,
    "per_unit": "cpu",
    "friendly_rate": "Hourly @ 2.0 + 0.0 per Cpu from 0.0 to Infinity",
    "created_on": "2020-04-23T15:00:29Z",
    "updated_on": "2020-05-28T13:24:33Z",
    "chargeback_rate_detail_measure_id": "1",
    "chargeback_rate_detail_currency_id": "5",
    "sub_metric": null
}
```

Deleting Chargeback Rates

Deleting Chargeback Rates can be done via either the **delete** post action or the DELETE HTTP method.

Request:

```
POST /api/rates/16

{
  "action" : "delete"
}
```

Response:

```
{
  "success": true,
  "message": "rates id: 16 deleting",
  "href": "http://localhost:3000/api/rates/16"
}
```

or simply:

```
DELETE /api/rates/16
```

Querying Assigned Chargeback Rates

Assigned chargeback rates on single rate:

```
GET /api/chargebacks/16?attributes=assigned_to
```

Response:

```
{
    "href": "http://localhost:3090/api/chargebacks/2",
    "id": "2",
    "guid": "7d7aaf20-5214-11df-a888-001d09066d98",
    "description": "Default",
    "rate_type": "Storage",
```

```
    "created_on": "2019-08-07T16:48:04Z",
    "updated_on": "2020-04-20T13:27:07Z",
    "default": true,
    "assigned_to": [
        {
            "tag": {
                "href": "http://localhost:3090/api/tags/48",
                "name": "finance",
                "category": "department"
            }
        },
        {
            "tag": {
                "href": "http://localhost:3090/api/tags/26",
                "name": "test",
                "category": "environment"
            }
        }
    ]
}
```

Assigned chargeback rates on collection:

```
GET /api/chargebacks?expand=resources&attributes=assigned_to
```

Response:

```
{
    "name": "chargebacks",
    "count": 10,
    "subcount": 2,
    "subquery_count": 2,
    "pages": 1,
    "resources": [
        {
            "href": "http://localhost:3090/api/chargebacks/1",
            "id": "1",
            "guid": "b47a0ef0-4335-11df-aba8-001d09066d98",
            "description": "Default",
            "rate_type": "Compute",
            "created_on": "2019-08-07T16:48:04Z",
            "updated_on": "2020-05-13T09:36:21Z",
            "default": true,
            "assigned_to": [
                {
                    "object": {
                        "id": "1",
                        "name": "Enterprise",
                        "description": "Enterprise Root Object",
                        "created_on": "2017-10-06T16:35:32Z",
                        "updated_on": "2017-10-06T16:35:32Z",
                        "settings": null
                    }
                }
            ]
        },
        {
            "href": "http://localhost:3090/api/chargebacks/2",
            "id": "2",
            "guid": "7d7aaf20-5214-11df-a888-001d09066d98",
            "description": "Default",
            "rate_type": "Storage",
            "created_on": "2019-08-07T16:48:04Z",
            "updated_on": "2020-05-13T09:36:21Z",
            "default": true,
            "assigned_to": [
                {
                    "object": {
                        "id": "31",
                        "type": "ManageIQ::Providers::Vmware::InfraManager::Storage",
                        ...

                    }
                },
                {
                    "object": {
                        "id": "19",
                        "type": "ManageIQ::Providers::Vmware::InfraManager::Storage"
                         ...
                    }
```

```
                },
                {
                    "object": {
                        "id": "27",
                        "type": "ManageIQ::Providers::Vmware::InfraManager::Storage"
                         ...
                    }
                },
                {
                    "object": {
                        "id": "14",
                        "type": "ManageIQ::Providers::Vmware::InfraManager::Storage"
                        ...
                    }
                }
            ]
        }
    ]
}
```

Assign action for Chargeback Rates

Assignments could be done for two rate types: “Compute” and “Storage”. There are few assignments type under each rates type:

“Compute” Rate Type

- Enterprise
- Selected Providers
- Selected Clusters
- Tagged VMs and Instances
- Tagged Container Images
- Labeled Container Images
- Tenants

“Storage” Rate Type

- Enterprise
- Selected Datastores
- Tagged Datastores
- Tenants

Example (chargebacks collection) - assign storage to chargeback rate with id 2 :

```
{
    "action" : "assign",
    "assignments" : [
        {
            "chargeback":
            {
                "id" : 2
            },
            "resource":
            {
                "href": "http://localhost:3090/api/data_stores/12"
            }
        }
    ]
}
```

Response:

```
{
    "success": true,
    "message": "Rates assigned successfully",
    "result": [
        {
            "chargeback": {
                "href": "http://localhost:3090/api/chargebacks/2",
                "description": "Default"
            },
```

```
            "resource": {
                "href": "http://localhost:3090/api/data_stores/12",
                "name": "EXPORT"
            }
        }
    ]
}
```

Basic Template for single chargeback request

```
POST /api/chargebacks/<chargeback_id>
```

```
{
    "action" : "assign",
    "assignments" : [
        {
            <resource>
        }
    ]
}
```

Basic Template for chargebacks collection request

```
POST /api/chargebacks
```

```
{
    "action" : "assign",
    "assignments" : [
        {
            "chargeback":
            {
                <chargeback>
            },
            <resource>
        }
    ]
}
```

<chargeback> is same for all types of assignments. For **<chargeback>** can be used **href** or **id**:

- `{"href": "http://localhost:3090/api/chargebacks/1"}`
- `{"id": "1"}`

<resource> can be different across types of assignments - formats are described in next section.

Enterprise

<resource>:

```
"href": "http://localhost:3090/api/enterprises/1"
```

```
POST /api/chargebacks
```

```
{
    "action" : "assign",
    "assignments" : [
        {
            "chargeback":
            {
                "id" : 2
            },
            "resource":
            {
                "href": "http://localhost:3090/api/enterprises/1"
            }
        }
    ]
}
```

Response

```
{
    "success": true,
    "message": "Rates assigned successfully",
    "result": [
        {
            "chargeback": {
                "href": "http://localhost:3090/api/chargebacks/2",
                "description": "Default"
            },
```

```
            "resource": {
                "href": "http://localhost:3090/api/enterprises/1",
                "name": "Enterprise"
            }
        }
    ]
}
```

Selected Providers

<resource>:

```
"href": "http://localhost:3090/api/enterprises/1"
```

Selected Clusters

<resource>:

```
"href": "http://localhost:3090/api/clusters/1"
```

Tagged VMs and Instances

Identification can be done by tag's name and tag's category, by **id** or by **href**.

assignment_prefix is string which partly determines type of assignment. In this case for "VMs and Instances" value for **assignment_prefix** is **vm**.

<resource>:

```
- "tag": {
            "category" : "department",
            "name" : "finance",
            "assignment_prefix": "vm"
        }
- "tag": {
            "id" : 28,
            "assignment_prefix": "vm"
        }
- "tag": {
            "href" : 'http://localhost:3090/api/tags/28,
            "assignment_prefix": "vm"
        }
```

Request

```
POST /api/chargebacks
```

```
{
  "action" : "assign",
  "assignments" : [
   {
       "chargeback": { "id" : 1},
       "tag": { "category" : "department", "name" : "finance", "assignment_prefix": "vm" }
    },
    {
      "chargeback": { "id" : 1},
      "tag": { "id" : 2}
    },
    {
      "chargeback": { "id" : 3},
      "tag": { "href" : "http://localhost:3090/api/tags/28", "assignment_prefix": "vm" }
    }
  ]
}
```

Tagged Container Images

Identification can be done by tag's name and tag's category, by **id** or by **href**.

assignment_prefix is string which partly determines type of assignment. - In this case for "Container Images" value for **assignment_prefix** is **container_image**

<resource>:

```
"tag":
{
    "category" : "department",
    "name" : "finance",
    "assignment_prefix": "container_image"
}
```

or

```
"tag":
{
    "id" : 28,
    "assignment_prefix": "container_image"
}
```

or

```
"tag":
{
    "href" : 'http://localhost:3090/api/tags/28,
    "assignment_prefix": "container_image"
}
```

Request

```
POST /api/chargebacks
```

```
{
  "action" : "assign",
  "assignments" : [
   {
       "chargeback": { "id" : 1},
       "tag": { "category" : "department", "name" : "finance", "assignment_prefix": "vm" }
    },
    {
      "chargeback": { "id" : 1},
      "tag": { "id" : 2}
    },
    {
      "chargeback": { "id" : 3},
      "tag": { "href" : "http://localhost:3090/api/tags/28", "assignment_prefix": "vm" }
    }
  ]
}
```

Labeled Container Images

```
<resource>:
```

```
"href": "http://localhost:3090/api/container_images/934/custom_attributes/30331"
```

This resource represents custom attributes on container images, so called “labels”.

Request

```
POST /api/chargebacks
```

```
{
  "action" : "assign",
    "assignments" :
    [
      {
        "chargeback": { "id" : 1},
        "resource": { "href": "http://localhost:3090/api/container_images/934/custom_attributes/30331" }
      },
      {
        "chargeback": { "id" : 1},
        "resource": { "href": "http://localhost:3090/api/container_images/934/custom_attributes/30324" }
      }
    ]
}
```

Tenants

```
<resource>:
```

```
"href": "http://localhost:3090/api/tenants/1"
```

Example

Request

```
POST /api/chargebacks
```

```
{
    "action" : "assign",
    "assignments" : [
        {
```

```
            "chargeback": {
                "id" : 2
            },
            "resource": {
                "href": "http://localhost:3090/api/tenants/1"
            }
        },
        {
            "chargeback": {
                "id" : 2
            },
            "resource": {
                "href": "http://localhost:3090/api/tenants/2"
            }
        }
    ]
}
```

Response

```
{
    "success": true,
    "message": "Rates assigned successfully",
    "result": [
        {
            "chargeback": {
                "href": "http://localhost:3090/api/chargebacks/2",
                "description": "Default"
            },
            "resource": {
                "href": "http://localhost:3090/api/tenants/1",
                "name": "My Company"
            }
        },
        {
            "chargeback": {
                "href": "http://localhost:3090/api/chargebacks/2",
                "description": "Default"
            },
            "resource": {
                "href": "http://localhost:3090/api/tenants/2",
                "name": "OpenStack Cloud Provider OpenStack"
            }
        }
    ]
}
```

Enterprise

<resource>:

```
"resource" :
{
    href": "http://localhost:3090/api/enterprises/1"
}
```

Request

```
POST /api/chargebacks

{
    "action" : "assign",
    "assignments" : [
        {
            "chargeback":
            {
                "id" : 2
            },
            "resource":
            {
                "href": "http://localhost:3090/api/enterprises/1"
            }
        }
    ]
}
```

Selected Datastores

<resource>:

```
"resource":
{
```

```
    "href": "http://localhost:3090/api/enterprises/1"
}
```

Example

Request

```
POST /api/chargebacks
```

```
{
    "action" : "assign",
    "assignments" : [
        {
            "chargeback" :
            {
                "id" : 2
            },
            "resource" :
            {
                "href" : "/api/data_stores/1"
            }
        }
    ]
}
```

Response

```
{
    "success": true,
    "message": "Rates assigned successfully",
    "result": [
        {
            "chargeback": {
                "href": "http://localhost:3090/api/chargebacks/2",
                "description": "Default"
            },
            "resource": {
                "href": "http://localhost:3090/api/data_stores/1",
                "name": "Shared-NFS"
            }
        }
    ]
}
```

Tagged Datastores

<resource>:

```
"tag":
{
    "href": "http://localhost:3090/api/tags/26",
    "assignment_prefix": "storage"
}
```

or

```
"tag":
{
    "category" : "department",
    "name" : "finance",
    "assignment_prefix": "storage"
}
```

or

```
"tag":
{
    "id" : "26",
    "assignment_prefix": "storage"
}
```

Example

Request

```
POST /api/chargebacks
```

```
{
    "action" : "assign",
    "assignments" : [
        {
            "tag": {
```

```
                "category" : "department",
                "name" : "finance",
                "assignment_prefix": "storage"
            }
        }
    ]
}
```

Response

```
{
    "success": true,
    "message": "Rates assigned successfully",
    "result": [
        {
            "chargeback": {
                "href": "http://localhost:3090/api/chargebacks/2",
                "description": "Default"
            },
            "tag": {
                "href": "http://localhost:3090/api/tags/48",
                "name": "finance",
                "category": "department",
                "assignment_prefix": "storage"
            }
        }
    ]
}
```

Tenants

<resource>:

```
"href": "http://localhost:3090/api/tenants/1"
```

Request

```
POST /api/chargebacks
```

```
{
    "action" : "assign",
    "assignments" : [
        {
            "chargeback": {
                "id" : 2
            },
            "resource": {
                "href": "http://localhost:3090/api/tenants/1"
            }
        },
        {
            "chargeback": {
                "id" : 2
            },
            "resource": {
                "href": "http://localhost:3090/api/tenants/2"
            }
        }
    ]
}
```

Response

```
{
    "success": true,
    "message": "Rates assigned successfully",
    "result": [
        {
            "chargeback": {
                "href": "http://localhost:3090/api/chargebacks/2",
                "description": "Default"
            },
            "resource": {
                "href": "http://localhost:3090/api/tenants/1",
                "name": "My Company"
            }
        },
        {
            "chargeback": {
                "href": "http://localhost:3090/api/chargebacks/2",
                "description": "Default"
            },
```

```
            "resource": {
                "href": "http://localhost:3090/api/tenants/2",
                "name": "OpenStack Cloud Provider OpenStack"
            }
        }
    ]
}
```

Unassign action for Chargeback Rates

Unassign action has same signature as assign action and action is ‘unassign’.

Example

Request

POST

```
{
  "action" : "unassign",
  "assignments" : [
     {
       "chargeback": { "id" : 3},
       "tag": { "category" : "prov_max_retirement_days", "name" : "30", "assignment_prefix": "vm" }
     }
  ]
}
```

Example of Chargeback Flow

- Create Chargeback Report
- Create Chargeback Rates
- Assign Chargeback Rates
- Run Chargeback Report
- Results of Chargeback Report

Create Chargeback Report

POST api/reports

```
{
  "action" : "import",
  "resource" : {
    "report" : {
        "rpt_group"   : "Custom",
        "rpt_type"    : "Custom",
        "menu_name"   : "Chargeback Flow Example",
        "col_order"   : ["vm_name", "display_range", ..],
        "cols"        : ["start_date", "display_range", "vm_name", ..],
        "headers"     : ["VM Name", "Date Range", ..],
        "title"       : "Chargeback Flow Example",
        "db"          : "ChargebackVm",
        "group"       : "y",
        "sortby"      : ["vm_name", "start_date"],
        "template_type" : "report",
        "order"          : "Ascending",
        "db_options" : {
            "rpt_type" : "ChargebackVm",
            "options"  : {
               <report_options>
            }
        }
    },
    "options" : { "save" : true }
  }
}
```

Attribute **"report"** contains JSON with report definition which can be obtained by exporting a chargeback report.

- **db** is type of report for chargeback purposes. Possible values are **ChargebackVm, ChargebackContainerProject, ChargebackContainerImage**.

Chargeback Report Options

Chargeback Report Options are present on index **options** in request json. There are also some options about filtering and grouping which are described in this section. Default attributes are described first and then there is explanation of attributes according to chargeback type of report(attribute **db**).

Example Request with **options**

```
POST /api/chargebacks
```

```
"db_options" : {
        "rpt_type" : "ChargebackVm",
        "options"  : {
            "interval" : "monthly",
            "interval_size" : 12,
            "end_interval_offset" : 1,
            "tenant_id" : "1",
            "method_for_allocated_metrics" : "max",
            "include_metrics:" : true,
            "cumulative_rate_calculation" : false,
            "groupby" : "date",
            "groupby_tag" : null,
            "groupby_label" : null
        }
}
...,
```

- **method_for_allocated_metrics** - **max** or **avg**
- **include_metrics** - **true** or **false** - enable/disable whether to include metrics data, when disable it uses current values of metric data or charged resources
- **cumulative_rate_calculation** - **true** or **false** enable/disable cumulative calculation for tagged resources
- **interval** is which time detail is used for the report. Possible values are **monthly, weekly, daily**

Metric data for chargeback report are taken from this interval:

```
`<Time.now-interval_size,  Time.now-end_interval_offset>`
```

- **interval_size, end_interval_offset** - attributes in **options**. there is used unit according to interval parameter, for example: when interval is “daily” then **interval_size=3** and **end_interval_offset=2** mean that generated report will be based on metrics starting 3 days ago to 1 day ago.

Request

```
POST /api/chargebacks
```

```
"report" : {
    "db_options" : {
        "options" : {
            ...
        }
    },
    ...
}
```

Filtering according to resource

This is part of json used in request below.

Each section lists possible options to filter according to resource’s type. There are used **ids** for identification each type of resource and and those **ids** are used in attributes according to type of resources. Name of attributes can be find in following sections. Attributes are located in json for request(example with tenant) on indexes **report**,
db_options and options:

Request

```
POST /api/chargebacks
```

```
"report" : {
    "db_options" : {
        "options" : {
            "tenant_id" : 1,
            ...
        }
    },
    ...
}
```

Chargeback for VMs (db=ChargebackVm)

tenant_id - Tenant id

or

owner - **userid** of user which owns VMs.

Request

```
POST /api/chargebacks
```

```
"report" : {
    "db_options" : {
        "options" : {
            "owner" : "admin",
            ...
        }
    },
    ...
}
```

or

tag - tag_path of tag

Request

```
POST /api/chargebacks
```

```
"report" : {
    "db_options" : {
        "options" : {
            "tag" : "/managed/prov_max_cpu/2",
            ...
        }
    },
    ...
}
```

Chargeback for Projects (db=ChargebackContainerProject)

tag - tag_path of tag

Request

```
POST /api/chargebacks
```

```
"report" : {
    "db_options" : {
        "options" : {
            "tag" : "/managed/prov_max_cpu/2",
            ...
        }
    },
    ...
}
```

or

provider_id - “all” or Openshift Provider id

entity_id - “all” or Container Project id

```
"report" : {
    "db_options" : {
        "options" : {
            "provider_id" : "41",
            "entity_id" : "21",
            ...
        }
    },
    ...
}
```

Chargeback for Container Images (db=ChargebackContainerImage)

provider_id - “all” or Openshift Provider id

entity_id - “all” or Container Image id

Request

```
POST /api/chargebacks
```

```
"report" : {
    "db_options" : {
        "options" : {
            "provider_id" : "all",
            "entity_id" : "all",
            ...
        }
    }
}
```

Grouping Report Options

Chargeback for VMs (db=ChargebackVm)

groupby - specifies a grouping option

Possible values for **groupby**:

date - grouping by date and Vms

date-only - grouping only by date

tenant - grouping by tenants

tag - name of tag category - attribute **groupby_tag** needs to be also specified. Values for **groupby_tag** are names of tag category which can be retrieved in **resources**

```
GET
/api/categories?expand=resources&attributes=name
```

Chargeback for Projects (db=ChargebackContainerProject)

groupby - specifies a grouping option

Possible values for **groupby**:

date - grouping by date and container projects

date-only - grouping only by date

project - grouping by container projects

tag - - name of tag category - attribute **groupby_tag** needs to be also specified. Values for **groupby_tag** are names of tag category which can be retrieved in **resources**

```
GET
/api/categories?expand=resources&attributes=name
```

Chargeback for Container Images (db=ChargebackContainerImage)

groupby - specifies a grouping option

Possible values for **groupby**:

date - grouping by date and containers

date-only - grouping only by date

project - grouping by container images

label - grouping by container labels and when this value is set - attribute **groupby_tag** needs to be also specified. Values for **groupby_tag** are names of labels which can be retrieved by **api/container_images/:id/custom_attributes** endpoint (attribute=name) for single container image.

Example request

```
GET
/api/container_images/934/custom_attributes?expand=resources&attributes=name&filter[]=section=docker_labels
```

Response

```
{
    "name": "custom_attributes",
    "count": 2041860,
    "subcount": 11,
    "subquery_count": 11,
    "pages": 1,
    "resources": [
        {
            "href": "http://localhost:3090/api/container_images/934/custom_attributes/30322",
            "name": "build-date",
            "id": "30322"
        },
        {
            "href": "http://localhost:3090/api/container_images/934/custom_attributes/30324",
            "name": "architecture",
            "id": "30324"
        }
    ],
    ...
}
```

Example of request to create chargeback report

Request

```
POST /api/chargebacks
```

```
{
  "action" : "import",
  "resource" : {
    "report" : {
        "rpt_group"   : "Custom",
        "rpt_type"    : "Custom",
        "menu_name"   : "Chargeback Flow Example 1",
        "col_order"   : ["vm_name", "display_range", "chargeback_rates", "memory_allocated_metric",
"memory_allocated_cost", "memory_allocated_rate", "memory_cost"],
        "cols"        : ["start_date", "display_range", "vm_name", "chargeback_rates",
"memory_allocated_metric", "memory_allocated_cost", "memory_allocated_rate", "memory_cost"],
        "headers"     : ["VM Name", "Date Range", "Chargeback Rates", "Memory Allocated over Time Period",
"Memory Allocated Cost" , "Memory Allocated Rate","Memory Total Cost"],
        "title"       : "Chargeback Flow Example 1",
        "db"          : "ChargebackVm",
        "group"       : "y",
        "sortby"      : ["vm_name", "start_date"],
        "template_type" : "report",
        "order"          : "Ascending",
        "db_options" : {
            "rpt_type" : "ChargebackVm",
            "options"  : {
                "interval" : "monthly",
                "interval_size" : 12,
                "end_interval_offset" : 1,
                "tenant_id" : "1",
                "method_for_allocated_metrics" : "max",
                "include_metrics:" : true,
                "cumulative_rate_calculation" : false,
                "groupby" : "date",
                "groupby_tag" : null,
                "groupby_label" : null
            }
        }
    },
    "options" : { "save" : true }
  }
}
```

Response

```
{
    "results": [
        {
            "success": true,
            "message": "Imported Report: [Chargeback Flow Example 1]",
            "result": {
                "id": "553",
                "name": "Chargeback Flow Example 1",
                "title": "Chargeback Flow Example 1",
                "rpt_group": "Custom",
                "rpt_type": "Custom",
                "priority": null,
                "db": "ChargebackVm",
```

```
                "cols": [
                    "start_date",
                    "display_range",
                    "vm_name",
                    "chargeback_rates",
                    "memory_allocated_metric",
                    "memory_allocated_cost",
                    "memory_allocated_rate",
                    "memory_cost"
                ],
                "include": null,
                "col_order": [
                    "vm_name",
                    "display_range",
                    "chargeback_rates",
                    "memory_allocated_metric",
                    "memory_allocated_cost",
                    "memory_allocated_rate",
                    "memory_cost"
                ],
                "headers": [
                    "VM Name",
                    "Date Range",
                    "Chargeback Rates",
                    "Memory Allocated over Time Period",
                    "Memory Allocated Cost",
                    "Memory Allocated Rate",
                    "Memory Total Cost"
                ],
                "conditions": null,
                "order": "Ascending",
                "sortby": [
                    "vm_name",
                    "start_date"
                ],
                "group": "y",
                "graph": null,
                "dims": null,
                "created_on": "2020-05-19T12:16:31Z",
                "updated_on": "2020-05-19T12:16:31Z",
                "filename": null,
                "file_mtime": null,
                "categories": null,
                "timeline": null,
                "template_type": "report",
                "where_clause": null,
                "db_options": {
                    "rpt_type": "ChargebackVm",
                    "options": {
                        "interval": "monthly",
                        "interval_size": 12,
                        "end_interval_offset": 1,
                        "tenant_id": "1",
                        "method_for_allocated_metrics": "max",
                        "include_metrics:": true,
                        "cumulative_rate_calculation": false,
                        "groupby": "date",
                        "groupby_tag": null,
                        "groupby_label": null
                    }
                },
                "generate_cols": null,
                "generate_rows": null,
                "col_formats": null,
                "tz": null,
                "time_profile_id": null,
                "display_filter": null,
                "col_options": null,
                "rpt_options": null,
                "miq_group_id": "2",
                "user_id": "1"
            }
        }
    ]
}
```

Create Chargeback Rates

Chargeback rate is complex structure. Create chargeback rate requires two endpoints: **/api/chargebacks** and **/api/rates**. Endpoint **/api/chargebacks** manage top level rate entity which aggregate other relations like rate details(**api/rates**), currencies(**api/currencies**),

measures(**api/measures**), chargeable_fields(**api/chargeable_fields**) and chargeback tiers. This entity is also used to assigning rates to any resource.

Create Chargebacks (top-level rate entity)

Request

```
POST /api/chargebacks
```

```
{
  "guid" : "daily",
  "description" : "1",
  "rate_type" : "Compute"
}
```

Response

```
{
    "results": [
        {
            "href": "http://localhost:3090/api/chargebacks/14",
            "id": "14",
            "guid": "daily",
            "description": "1",
            "rate_type": "Compute",
            "created_on": "2020-05-26T12:15:51Z",
            "updated_on": "2020-05-26T12:15:51Z",
            "default": false
        }
    ]
}
```

Now we need to create rate details(**api/rates**) which includes chargebable_fields, currenies, measures and tiers. It is also important assign rate details to chargeback top level entity by **chargeback_rate** attribute.

Request

```
POST /api/rates
```

```
{
  "per_time" : "daily",
  "chargeback_rate" : {"href": "http://localhost:3090/api/chargebacks/14"},
  "chargeable_field_id" : "1",
  "description": "My CPU allocation rate",
  "group" : "cpu",
  "per_unit" : "megahertz",
  "source" : "allocated",
  "chargeback_rate_detail_measure_id": 1,
  "detail_currency" : {"id": 5},
  "detail_measure" : {"href": "http://localhost:3090/api/measures/3"},
  "chargeable_field" : {"href": "http://localhost:3090/api/chargeable_fields/3"},
  "chargeback_tiers" : [
    {
      "start": 0.0,
      "finish":1,
      "fixed_rate": 2.0,
      "variable_rate": 0.0
    },
    {
      "start": 1,
      "finish": "infinity",
      "fixed_rate": 2.0,
      "variable_rate": 0.0
    }
  ]
}
```

Response

```
{
    "results": [
        {
            "href": "http://localhost:3090/api/rates/1862",
            "id": "1862",
            "enabled": true,
            "description": "My CPU allocation rate",
            "group": "cpu",
            "source": "allocated",
            "metric": null,
            "per_time": "daily",
            "per_unit": "megahertz",
```

```
            "friendly_rate": "Daily @ 2.0 + 0.0 per Megahertz from 0.0 to 1.0\nDaily @ 2.0 + 0.0 per
Megahertz from 1.0 to Infinity",
            "chargeback_rate_id": "14",
            "created_on": "2020-05-26T12:36:32Z",
            "updated_on": "2020-05-26T12:36:32Z",
            "chargeback_rate_detail_measure_id": "3",
            "chargeback_rate_detail_currency_id": "5",
            "chargeable_field_id": "3",
            "sub_metric": null
        }
    ]
}
```

Assign Chargeback Rate

There are examples in those sections:

- Querying Assigned Chargeback Rates
- Assign action for Chargeback Rates
- Unassign action for Chargeback Rates

Run Chargeback Report

Please refer to Run Report section for details on how to run chargeback report. Running report requires report id or report href:

Report id or report href can be obtained by chargeback report creation or by listing chargeback reports:

Request

```
GET /api/reports?expand=resources&attributes=db&filter[]=db='ChargebackVm'&filter[]=or
db='ChargebackContainer%'
```

Response

```
{
    "name": "reports",
    "count": 519,
    "subcount": 47,
    "subquery_count": 47,
    "pages": 1,
    "resources": [
        {
            "href": "http://localhost:3090/api/reports/157",
            "db": "ChargebackContainerImage",
            "id": "157"
        },
        {
            "href": "http://localhost:3090/api/reports/152",
            "db": "ChargebackVm",
            "id": "152"
        },
...
}
```

Results of Chargeback Report

There is request which list of report result.

Allowed parameters:

`hash_attribute=value` - enables hash sorting and pagination for value column

`sort_by` - column from **`MiqReportResult#report#col_order`**

`sort_order` - **`asc`** or **`desc`**

`limit` - count of records for one page

`offset` - result will start form **`<offset>`**. record

`filter_column` - chosen column for filtering

`filter_string` - this string value is used for filtering on **`<filter_column>`**, record of report result will be returned in API when **`<filter_string>`** is contained in value of **`<filter_column>`** .

There is possibility to filter according to more columns. We need to add other filter parameters with suffix **`_X`** where **`X`** is number (**`..&filter_column_1=name&filter_string_1=TIE&filter_column_2=hostname&filter_string_2=test`**)

Request

```
GET /api/results/23039?hash_attribute=result_set&sort_by=&sort_order=&limit=20&offset=0&filter_string=windemo
```

Response


```
{
    "href": "http://localhost:3090/api/results/23039",
    "id": "23039",
    "name": "Chargeback - Demo",
    ...,
    "result_set": [
        {
            "start_date": "01/01/19 00:00:00 UTC",
            "display_range": "Jan 2019",
            "vm_name": "windemo",
            "owner_name": "Cloud Operators",
            "provider_name": "vCenter",
            "total_cost": "$48,303.51",
            "cpu_cost": "$935.65",
            "cpu_used_cost": "$191.65",
            "cpu_used_metric": "12.88 MHz",
            "memory_cost": "$1,609.69",
            "memory_used_cost": "$1,609.69",
            "memory_used_metric": "108.18 MB",
            "net_io_used_cost": "$372.00",
            "net_io_used_metric": "0.00 KBps",
            "chargeback_rates": "Default"
        },
        {
            "start_date": "02/01/19 00:00:00 UTC",
            "display_range": "Feb 2019",
            "vm_name": "windemo",
            "owner_name": "Cloud Operators",
            "provider_name": "vCenter",
            "total_cost": "$43,851.09",
            "cpu_cost": "$861.44",
            "cpu_used_cost": "$189.44",
            "cpu_used_metric": "14.09 MHz",
            "memory_cost": "$1,659.73",
            "memory_used_cost": "$1,659.73",
            "memory_used_metric": "123.49 MB",
            "net_io_used_cost": "$336.00",
            "net_io_used_metric": "0.00 KBps",
            "chargeback_rates": "Default"
        },
        {
            "start_date": "03/01/19 00:00:00 UTC",
            "display_range": "Mar 2019",
            "vm_name": "windemo",
            "owner_name": "Cloud Operators",
            "provider_name": "vCenter",
            "total_cost": "$28,906.99",
            "cpu_cost": "$553.43",
            "cpu_used_cost": "$113.43",
            "cpu_used_metric": "12.89 MHz",
            "memory_cost": "$1,231.94",
            "memory_used_cost": "$1,231.94",
            "memory_used_metric": "139.99 MB",
            "net_io_used_cost": "$220.00",
            "net_io_used_metric": "0.00 KBps",
            "chargeback_rates": "Default"
        }
    ],
    "count": 3,
    "subcount": 3,
    "pages": 1
}
```


- Cloud Volumes
  - Querying Cloud Volumes
  - Deleting Cloud Volumes

Cloud Volumes

Management of Cloud Volumes is provided via the following collection:

```
/api/cloud_volumes
```

The following actions are available on cloud_volumes:

- Querying Cloud Volumes
- Deleting Cloud Volumes

Querying Cloud Volumes

Querying all cloud volumes in the system is simply:

```
GET /api/cloud_volumes
```

And getting details of a specific Cloud Volume is as follows:

```
GET /api/cloud_volumes/:id
```

Deleting Cloud Volumes

Cloud Volumes can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/cloud_volumes/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/cloud_volumes/101
```

Deleting multiple Cloud Volumes can be done as follows:

```
POST /api/cloud_volumes
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/cloud_volumes/101" },
    { "href" : "http://localhost:3000/api/cloud_volumes/102" },
    ...
  ]
}
```

- Conditions
  - Querying Conditions
  - Creating Conditions
  - Editing Conditions
  - Deleting Conditions

Conditions

Management of conditions is provided via the following collection:

```
/api/conditions
```

Full CRUD actions on conditions is available:

- Querying Conditions
- Creating Conditions
- Editing Conditions
- Deleting Conditions

Querying Conditions

Querying all conditions in the system is simply:

```
GET /api/conditions
```

Getting details on a specific Condition:

```
GET /api/conditions/:id
```

Creating Conditions

Conditions can be created via a POST to the conditions collection or via the **create** action signature which also allows creation of multiple conditions in a single request.

```
POST /api/conditions
```

```
{
  "name" : "sample_condition",
  "description" : "Sample Condition",
  "expression" : { "=" : { "field" : "ExtManagementSystem.container_images-architecture", "value" : "dsa" }
},
  "towhat" : "ExtManagementSystem",
  "modifier" : "allow"
}
```

or creating multiple conditions:

```
{
  "action" : "create",
  "resources" : [
    { "name" : "condition_1", "description" : "Condition 1", ... },
    { "name" : "condition_2", "description" : "Condition 2", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating conditions.

Editing Conditions

```
POST /api/conditions/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "Updated Condition Description"
  }
}
```

or editing multiple conditions:

```
POST /api/conditions
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/conditions/101",
      "description" : "Updated Condition Description 1"
    },
    {
      "href" : "http://localhost:3000/api/conditions/102",
      "description" : "Updated Condition Description 2"
    },
    ...
  ]
}
```

Deleting Conditions

Conditions can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/conditions/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/conditions/101
```

Deleting multiple conditions can be done as follows:

```
POST /api/conditions
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/conditions/101" },
```

```
    { "href" : "http://localhost:3000/api/conditions/102" },
    ...
  ]
}
```


- Configuration Script Sources
  - Querying Configuration Script Sources
  - Creating Configuration Script Sources
  - Editing Configuration Script Sources
  - Refreshing Configuration Script Sources
  - Request:
  - Response:
  - Deleting Configuration Script Sources

Configuration Script Sources

Management of configuration script sources is provided via the following collection:

```
/api/configuration_script_sources
```

Full CRUD actions on configuration script sources is available:

- Querying Configuration Script Sources
- Creating Configuration Script Sources
- Editing Configuration Script Sources
- Refreshing Configuration Script Sources
- Deleting Configuration Script Sources

Querying Configuration Script Sources

Querying all configuration script sources in the system is simply:

```
GET /api/configuration_script_sources
```

Getting details on a specific configuration script source:

```
GET /api/configuration_script_sources/:id
```

Querying configuration script payloads of a particular configuration script source:

```
GET /api/configuration_script_sources/:id/configuration_script_payloads?expand=resources
```

Querying the configuration script payloads of a particular configuration script source that belong to a specific region:

```
GET /api/configuration_script_sources/:id/configuration_script_payloads?
expand=resources&filter[]=region_number=10
```

Creating Configuration Script Sources

Configuration script sources can be created via a POST to the configuration script sources collection or via the **create** action signature which also allows creation of multiple configuration script sources in a single request.

```
POST /api/configuration_script_sources
```

```
{
  "name" : "My Project",
  "description" : "My Project's Description",
  "manager_resource" : { "href" : "http://localhost:3000/api/providers/7" },
  "related" : {}
}
```

or creating multiple configuration script sources:

```
{
  "action" : "create",
  "resources" : [
    { "name" : "My First Project", "description" : "My First Project Description", ... },
    { "name" : "My Second Project", "description" : "My Second Project Description", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating configuration script sources.

Editing Configuration Script Sources

```
POST /api/configuration_script_sources/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "Updated Project Description"
  }
}
```

or editing multiple configuration script sources:

```
POST /api/configuration_script_sources
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/configuration_script_sources/101",
      "description" : "Updated Project Description 1"
    },
    {
      "href" : "http://localhost:3000/api/configuration_script_sources/102",
      "description" : "Updated Project Description 2"
    },
    ...
  ]
}
```

Refreshing Configuration Script Sources

Configuration script sources can be refreshed by posting a **refresh** action to a single resource or to multiple resources in bulk by targeting the collection.

```
POST /api/configuration_script_sources/:id
```

```
{
  "action" : "refresh"
}
```

or refreshing multiple configuration script sources:

```
POST /api/configuration_script_sources
```

```
{
  "action" : "refresh",
  "resources" : [
    { "id" : "101" },
    { "id" : "102" }
  ]
}
```

Example:

Request:

```
POST /api/configuration_script_sources/101
```

```
{
  "action" : "refresh"
}
```

Response:

```
{
  "success" : true,
  "message" : "Refreshing ConfigurationScriptSource id:101 name:'Test Configuration Script Source'",
  "task_id" : "7",
  "task_href" : "http://localhost:3000/api/tasks/7",
  "tasks" : [
    {
      "id" : "7",
      "href" : "http://localhost:3000/api/tasks/7"
    }
  ]
}
```

Deleting Configuration Script Sources

Configuration script sources can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/configuration_script_sources/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/configuration_script_sources/101
```

Deleting multiple configuration script sources can be done as follows:

```
POST /api/configuration_script_sources
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/configuration_script_sources/101" },
    { "href" : "http://localhost:3000/api/configuration_script_sources/102" },
    ...
  ]
}
```


- Custom Actions
  - Request:
  - Response:
    - Request:
    - Response:

Custom Actions

With button groups and custom buttons defined on records, the REST API provides visibility to custom actions that represent the custom buttons for the related resources. The custom actions are not returned when querying collections, they are only available when fetching an individual resource, e.g. /api/:collection/:id

This is made available with the *custom_actions* and *custom_action_buttons* virtual attributes on resources for the following collections:

/api/cloud_tenants
/api/cloud_volumes
/api/clusters
/api/container_nodes
/api/data_stores
/api/generic_objects
/api/hosts
/api/providers
/api/templates
/api/vms

- *custom_actions* return the custom buttons as laid out in the UI on the resource. The information returned includes any Resource Actions and related Dialogs for the custom button.
- *custom_action_buttons* return the array of custom buttons without the custom button groups hierarchy or details on related dialogs.

Fetching custom actions and custom action buttons for a resource:

```
GET /api/:collection/:id?attributes=custom_actions,custom_action_buttons
```

Example VM with custom buttons:

Request:

```
GET /api/vms/173?attributes=custom_actions,custom_action_buttons
```

Response:

```
{
  "href": "https://172.16.30.129/api/vms/173",
  "id": "173",
```

```
  "vendor": "vmware",
  "name": "aab-ldap",
  "location": "aab-ldap/aab-ldap.vmx",
  "host_id": "4",
  "created_on": "2018-01-09T20:23:37Z",
  "updated_on": "2018-01-09T20:23:37Z",
  "storage_id": "8",
  "guid": "dee6d969-7c69-44b6-bcd3-3b14f21702c5",
  "ems_id": "1",
  "uid_ems": "420c8f0d-4a8f-0b6d-fc3a-0c3cfab7b33f",
  "tools_status": "toolsNotRunning",
  "standby_action": "checkpoint",
  "power_state": "off",
  ...
  "custom_actions": {
    "buttons": [

    ],
    "button_groups": [
      {
        "id": "23",
        "name": "VM Actions|Vm|",
        "description": "VM Custom Actions",
        "created_on": "2018-01-09T20:36:50Z",
        "updated_on": "2018-01-09T20:41:40Z",
        "guid": "559b37d1-3232-4c51-8668-f994a5b9149e",
        "set_data": {
          "button_order": [
            1,
            2
          ],
          "button_icon": "pficon pficon-virtual-machine",
          "display": true,
          "applies_to_class": "Vm",
          "group_index": 1
        },
        "buttons": [
          {
            "id": "1",
            "guid": "22b5bf4f-d5d6-43af-bc7b-77460d120065",
            "description": "Analyze Virtual Machine",
            "applies_to_class": "Vm",
            "options": {
              "button_icon": "ff ff-timeline",
              "button_type": "default",
              "display": true,
              "open_url": false,
              "display_for": "single",
              "submit_how": "one"
            },
            "userid": "admin",
            "created_on": "2018-01-09T20:40:11Z",
            "updated_on": "2018-01-09T20:40:11Z",
            "name": "Analyze",
            "visibility": {
              "roles": [
                "_ALL_"
              ]
            },
            "resource_action": {
              "id": "21",
              "resource_id": "1",
              "resource_type": "CustomButton",
              "created_at": "2018-01-09T20:40:11Z",
              "updated_at": "2018-01-09T20:40:11Z",
              "ae_namespace": "SYSTEM",
              "ae_class": "PROCESS",
              "ae_instance": "Automation",
              "ae_attributes": {
                "request": "general/process/analyze"
              }
            },
            "enabled": true,
            "href": "https://172.16.30.129/api/custom_buttons/1"
          },
          {
            "id": "2",
            "guid": "e9e77bb3-f52a-4d08-8d58-13923031c6de",
            "description": "Ping Virtual Machine",
            "applies_to_class": "Vm",
            "options": {
```

```
            "button_icon": "ff ff-scsi",
            "button_color": "#b4c10f",
            "button_type": "default",
            "display": true,
            "open_url": false,
            "display_for": "single",
            "submit_how": "one"
          },
          "userid": "admin",
          "created_on": "2018-01-09T20:41:40Z",
          "updated_on": "2018-01-09T20:41:40Z",
          "name": "Ping",
          "visibility": {
            "roles": [
              "_ALL_"
            ]
          },
          "resource_action": {
            "id": "22",
            "resource_id": "2",
            "resource_type": "CustomButton",
            "created_at": "2018-01-09T20:41:40Z",
            "updated_at": "2018-01-09T20:41:40Z",
            "ae_namespace": "SYSTEM",
            "ae_class": "PROCESS",
            "ae_instance": "Automation",
            "ae_attributes": {
              "request": "general/virtual_machines/ping"
            }
          },
          "enabled": true,
          "href": "https://172.16.30.129/api/custom_buttons/2"
        }
      ],
      "href": "https://172.16.30.129/api/custom_button_sets/23"
    }
  ]
},
"custom_action_buttons": [
  {
    "href": "https://172.16.30.129/api/custom_buttons/1",
    "id": "1",
    "guid": "22b5bf4f-d5d6-43af-bc7b-77460d120065",
    "description": "Analyze Virtual Machine",
    "applies_to_class": "Vm",
    "options": {
      "button_icon": "ff ff-timeline",
      "button_type": "default",
      "display": true,
      "open_url": false,
      "display_for": "single",
      "submit_how": "one"
    },
    "userid": "admin",
    "created_on": "2018-01-09T20:40:11Z",
    "updated_on": "2018-01-09T20:40:11Z",
    "name": "Analyze",
    "visibility": {
      "roles": [
        "_ALL_"
      ]
    }
  },
  {
    "href": "https://172.16.30.129/api/custom_buttons/2",
    "id": "2",
    "guid": "e9e77bb3-f52a-4d08-8d58-13923031c6de",
    "description": "Ping Virtual Machine",
    "applies_to_class": "Vm",
    "options": {
      "button_icon": "ff ff-scsi",
      "button_color": "#b4c10f",
      "button_type": "default",
      "display": true,
      "open_url": false,
      "display_for": "single",
      "submit_how": "one"
    },
    "userid": "admin",
    "created_on": "2018-01-09T20:41:40Z",
    "updated_on": "2018-01-09T20:41:40Z",
```

```
      "name": "Ping",
      "visibility": {
        "roles": [
          "_ALL_"
        ]
      }
    }
  ],
  "actions": [
    ...
  ]
}
```

When custom action buttons are defined on a VM, the returned VM resource is extended with custom actions which can be triggered via a POST action.

The action name to trigger the action is the lower-cased name of the custom button.

As an example, here's a VM without custom buttons so all actions returned are the common ones defined by the system:

```
{
  "href": "https://172.16.30.129/api/vms/172",
  "id": "172",
  "vendor": "vmware",
  "name": "aab-idp",
  "location": "aab-idp/aab-idp.vmx",
  "host_id": "4",
  "created_on": "2018-01-09T19:23:37Z",
  "updated_on": "2018-01-09T19:23:37Z",
  "storage_id": "8",
  ...
  "actions": [
    {
      "name": "edit",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    {
      "name": "add_event",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    ...
    {
      "name": "set_miq_server",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    {
      "name": "scan",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "https://172.16.30.129/api/vms/173"
    }
  ]
}
```

Here is an example of a VM with the additional custom buttons *analyze* and *ping* defined:

```
{
  "href": "https://172.16.30.129/api/vms/173",
  "id": "173",
  "vendor": "vmware",
  "name": "aab-ldap",
  "location": "aab-ldap/aab-ldap.vmx",
  "host_id": "4",
  "created_on": "2018-01-09T20:23:37Z",
  "updated_on": "2018-01-09T20:23:37Z",
  "storage_id": "8",
  "guid": "dee6d969-7c69-44b6-bcd3-3b14f21702c5",
  "ems_id": "1",
  "uid_ems": "420c8f0d-4a8f-0b6d-fc3a-0c3cfab7b33f",
  "tools_status": "toolsNotRunning",
  "standby_action": "checkpoint",
  "power_state": "off",
  ...
  "actions": [
```

```
    {
      "name": "edit",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    {
      "name": "add_event",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    ...
    {
      "name": "set_miq_server",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    {
      "name": "scan",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "https://172.16.30.129/api/vms/173"
    },
    {
      "name": "analyze",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    },
    {
      "name": "ping",
      "method": "post",
      "href": "https://172.16.30.129/api/vms/173"
    }
  ]
}
```

Here, the *analyze* and *ping* are custom buttons for the VM and are returned after the common actions.

For example, to request the *analyze* custom action shown above, this is simply a POST action as follows:

Request:

```
POST /api/vms/173

{
  "action" : "analyze"
}
```

Response:

Responses from custom actions are the normal action responses from the REST API:

```
{
  "success": true,
  "message": "Invoked custom action analyze for VM id: 173",
  "href": "http://localhost:3000/api/vms/173"
}
```


- Custom Attributes
  - Querying Custom Attributes
  - Specifying Attribute Actions
  - Custom Attribute Actions
  - Adding Custom Attributes
  - Editing Custom Attributes
  - Deleting Custom Attributes

Custom Attributes

Custom Attributes can be managed for several types of resources. This is done via the custom_attributes sub-collection as follows:

Resource Type	Custom Attribute Subcollection
Virtual Machines	/api/vms/:id/custom_attributes
Cloud Instances	/api/instances/:id/custom_attributes

Resource Type	Custom Attribute Subcollection
Providers	/api/providers/:id/custom_attributes
Services	/api/services/:id/custom_attributes

- Querying Custom Attributes
- Specifying Custom Attributes
- Custom Attribute Actions
- Adding Custom Attributes
- Editing Custom Attributes
- Deleting Custom Attributes

Querying Custom Attributes

Custom attributes of a resource can be asked for by expanding the subcollection as follows:

```
GET /api/vms/:id?expand=custom_attributes
GET /api/instances/:id?expand=custom_attributes
GET /api/providers/:id?expand=custom_attributes
GET /api/services/:id?expand=custom_attributes
```

Optionally, just the custom attributes of a resource can be queried:

```
GET /api/vms/:id/custom_attributes
GET /api/instances/:id/custom_attributes
GET /api/providers/:id/custom_attributes
GET /api/services/:id/custom_attributes
```

Specifying Attribute Actions

Custom attributes are specified as follows:

```
{
  "name" : "...",
  "value" : "...",
  "field_type" : "...",
  "source" : "...",
  "section" : "...",
}
```

For adding custom attributes, **name** field is required.

field_type, **source** and **section** are optional.

There is no default value for **field_type**. Allowed values for **field_type** are **DateTime**, **Date** and **Time**.

Default value for **source** is **EVM**. If **source** is set to **VC**, the custom attribute field will be updated on the provider.

Default value for **section** is **metadata**. Allowed values for **section** include **metadata** and **cluster_settings**.

For editing a custom attribute, specify the custom attribute either by **name**:

```
{
  "name" : "...",
  "value" : "..."
}
```

Or by **href**:

```
{
  "href" : "http://localhost:3000/api/vms/:id/custom_attributes/:id",
  "value" : "..."
}
```

Custom Attribute Actions

Action	Description
add	Adding Custom Attributes to a Resource
edit	Editing Custom Attributes of a Resource
delete	Deleting Custom Attributes from a Resource

Adding Custom Attributes

Adding custom attributes to a Resource:

```
POST /api/vms/:id/custom_attributes
POST /api/instances/:id/custom_attributes
POST /api/providers/:id/custom_attributes
POST /api/services/:id/custom_attributes

{
  "action" : "add",
  "resources" : [
    { ... },
    { ... },
    ...
  ]
}
```

For example, adding two custom attributes:

```
{
  "action" : "add",
  "resources" : [
    { "name" : "ca1", "value" : "value 1" },
    { "name" : "ca2", "value" : "value 2" }
  ]
}
```

Editing Custom Attributes

Editing custom attributes of a Resource:

```
POST /api/vms/:id/custom_attributes
POST /api/instances/:id/custom_attributes
POST /api/providers/:id/custom_attributes
POST /api/services/:id/custom_attributes

{
  "action" : "edit",
  "resources" : [
    { ... },
    { ... },
    ...
  ]
}
```

For example:

```
{
  "action" : "edit",
  "resources" : [
    { "name" : "ca_name1", "value" : "updated custom attribute value 1" },
    { "name" : "ca_name2", "value" : "updated custom attribute value 2" }
  ]
}
```

Deleting Custom Attributes

Deleting custom attributes of a Resource:

```
POST /api/vms/:id/custom_attributes
POST /api/instances/:id/custom_attributes
POST /api/providers/:id/custom_attributes
POST /api/services/:id/custom_attributes

{
  "action" : "delete",
  "resources" : [
    { ... },
    { ... },
    ...
  ]
}
```

In the following example, a request to delete custom attributes of a VM by name and by href:

```
{
  "action" : "delete",
  "resources" : [
    { "name" : "ca_name1" },
    { "href" : "http://localhost:3000/api/vms/:id/custom_attributes/:id" }
  ]
}
```

Custom attributes can be deleted via a *delete* action as follows:

```
POST /api/vms/:id/custom_attributes/:ca_id
POST /api/instances/:id/custom_attributes/:ca_id
POST /api/providers/:id/custom_attributes/:ca_id
POST /api/services/:id/custom_attributes/:ca_id
```

```
{
  "action" : "delete"
}
```

They can also be deleted via the DELETE method:

```
DELETE /api/vms/:id/custom_attributes/:ca_id
DELETE /api/instances/:id/custom_attributes/:ca_id
DELETE /api/providers/:id/custom_attributes/:ca_id
DELETE /api/services/:id/custom_attributes/:ca_id
```


- Custom Button Management
  - Managing Custom Button Sets
    - Querying Custom Button Sets
    - Creating Custom Button Sets
    - Editing Custom Button Sets
    - Deleting Custom Button Sets
  - Managing Custom Buttons
    - Querying Custom Buttons
    - Creating Custom Buttons
    - Editing Custom Buttons
    - Deleting Custom Buttons

Custom Button Management

Management of Custom Buttons is provided with the following collections:

```
/api/custom_button_sets
/api/custom_buttons
```


- Managing Custom Button Sets
  - Querying Custom Button Sets
  - Creating Custom Button Sets
  - Editing Custom Button Sets
  - Deleting Custom Button Sets
- Managing Custom Buttons
  - Querying Custom Buttons
  - Creating Custom Buttons
  - Editing Custom Buttons
  - Deleting Custom Buttons

Managing Custom Button Sets

Querying Custom Button Sets

Querying all available Custom Button Sets:

```
GET /api/custom_button_sets
```

Getting details on a specific Custom Button Set:

```
GET /api/custom_button_sets/:id
```

Creating Custom Button Sets

Custom Button Sets can be created as follows:

```
POST /api/custom_button_sets
```

```
{
  "name"        : "Generic Object Custom Button Group",
  "description" : "Generic Object Custom Button Group Description",
  "set_data"    : {
    "button_icon"      : "ff ff-view-expanded",
    "button_color"     : "#4727ff",
    "display"          : true,
    "applies_to_class" : "GenericObjectDefinition",
    "applies_to_id"    : "50"
  }
}
```

Editing Custom Button Sets

Updating Custom Button Sets can be done via the *edit* action as follows:

```
POST /api/custom_button_sets/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "name" : "Updated Custom Button Group",
    "description" : "Updated Custom Button Group Description",
    "set_data" : {
      "button_icon" : "ff ff-closed"
    }
  }
}
```

Custom Button Sets can also be updated in bulk:

```
POST /api/custom_button_sets
```

```
{
  "action" : "edit",
  "resources" : [
    { "id" : "11", "set_data" : { "button_icon" : "ff ff-closed" } },
    { "id" : "12", "set_data" : { "button_icon" : "ff ff-closed" } },
    { "id" : "13", "set_data" : { "button_icon" : "ff ff-closed" } }
  ]
}
```

Custom button sets can also be updated with the PUT method as follows:

```
PUT /api/custom_button_sets/:id
```

```
{
  "name" : "Updated Custom Button Group",
  "description" : "Updated Custom Button Group Description",
  "set_data" : {
    "button_icon" : "ff ff-closed"
  }
}
```

Deleting Custom Button Sets

Custom Button Sets can be deleted via the *delete* action as follows:

```
POST /api/custom_button_sets/:id
```

```
{
  "action" : "delete"
}
```

and in bulk as follows:

```
POST /api/custom_button_sets
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/custom_button_sets/21" },
    { "href" : "http://localhost:3000/api/custom_button_sets/22" },
    { "id" : "23" }
  ]
}
```

A single Custom Button Set can also be deleted with the DELETE method as follows:

```
DELETE /api/custom_button_sets/:id
```

Managing Custom Buttons

Querying Custom Buttons

Querying all available Custom Buttons:

```
GET /api/custom_buttons
```

Getting details on a specific Custom Button:

```
GET /api/custom_buttons/:id
```

Creating Custom Buttons

Custom Buttons can be created as follows:

```
POST /api/custom_buttons
```

```
{
  "name"              : "Generic Object Custom Button",
  "description"       : "Generic Object Custom Button Description",
  "applies_to_class" : "GenericObjectDefinition",
  "options"           : {
    "button_icon"     : "ff ff-view-expanded",
    "button_color"    : "#4727ff",
    "display"         : true
  },
  "resource_action"   : {
    "ae_namespace"    : "SYSTEM",
    "ae_class"        : "PROCESS",
    "dialog_id"       : "1"
  },
  "visibility"        : {
    "roles"           : ["_ALL_"]
  }
}
```

Editing Custom Buttons

Updating Custom Buttons can be done via the *edit* action as follows:

```
POST /api/custom_buttons/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "name" : "Updated Custom Button",
    "description" : "Updated Custom Button Description",
    "resource_action" : {
      "ae_namespace"  : "SYSTEM2"
    }
  }
}
```

Custom Buttons can also be updated in Bulk as follows:

```
POST /api/custom_buttons
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/custom_buttons/31",
      "options" :  { "display" : false }
    },
    {
      "href" : "http://localhost:3000/api/custom_buttons/32",
      "options" :  { "display" : false }
    },
    {
      "id"   : "33",
      "options" :  { "display" : false }
    }
  ]
}
```

Deleting Custom Buttons

Custom Buttons can be deleted via the *delete* action as follows:

```
POST /api/custom_buttons/:id

{
  "action" : "delete"
}
```

Multiple Custom Buttons can also be deleted in a single request as follows:

```
POST /api/custom_buttons

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/custom_buttons/21" },
    { "id" : "22" },
    { "id" : "23" }
  ]
}
```

Single Custom Buttons can also be deleted with the DELETE method as follows:

```
DELETE /api/custom_buttons/:id
```


- Event Streams
  - Filtering by event type
  - Filtering by timestamp
  - Filtering by target type
  - Filtering by target type and id

Event Streams

Querying of Event Streams is made available via the following collection:

```
/api/event_streams
```

Event Streams can be queried as follows:

```
GET /api/event_streams
```

An individual Event Stream can also be fetched as follows:

```
GET /api/event_streams/:id
```

Relevant filter[] parameters when Event Streams are queried:

Parameter	Examples
event_type	event_type=vm_start, event_type=vm_stop
timestamp	timestamp>2017-01-03, timestamp<2017-12-31
target_type	target_type=VmOrTemplate
target_id	target_id=11

Filtering by event type

```
GET /api/event_streams?filter[]=event_type=vm_start
```

Filtering by timestamp

```
GET /api/event_streams?filter[]=timestamp>2017-01-03
                      &filter[]=timestamp<2017-01-10
```

Filtering by target type

```
GET /api/event_streams?filter[]=target_type=VmOrTemplate
```

Filtering by target type and id

```
GET /api/event_streams?filter[]=target_type=VmOrTemplate
                      &filter[]=target_id=21
```


- Flavor Management
  - Querying Flavors
  - Managing Flavors
  - Create Flavors
  - Delete Flavors

Flavor Management

Management of Flavors is available via the following endpoint:

```
/api/flavors
```

Querying Flavors

Querying all available Flavors:

```
GET /api/flavors
```

Getting details on a specific Flavor:

```
GET /api/flavors/:id
```

Querying the Flavors for a specific Provider:

```
GET /api/providers/:id/flavors
```

Getting details on a specific Flavor of a Provider:

```
GET /api/providers/:id/flavors/:flavor_id
```

Managing Flavors

The following actions are available on Flavors of Providers:

Action	Description
create	Create Flavors
delete	Delete Flavors

Create Flavors

Flavors can be created directly for a Provider as follows:

```
POST /api/providers/:id/flavors
```

```
{
  "name" : "base-system",
  "cpu_cores" : 2,
  "memory" :  4096000000,
  "root_disk_size" : 16384000000
}
```

Delete Flavors

Flavors of a Provider can be deleted via the *delete* action:

```
POST /api/providers/:id/flavors/:flavor_id
```

```
{
  "action" : "delete"
}
```

Multiple Flavors can also be deleted in a single request as follows:

```
POST /api/providers/:id/flavors
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/flavors/6" },
    { "id" : "7" }
  ]
}
```

Single Flavors can also be deleted with the DELETE HTTP method as follows:

```
DELETE /api/providers/:id/flavors/:flavor_id
```

- Generic Object Management
  - Generic Object Definitions
    - Querying Generic Object Definitions
    - Creating Generic Object Definitions
    - Editing Generic Object Definitions
      - Add Attributes

        - Remove Attributes
        - Add Methods
        - Remove Methods
        - Add Associations
        - Remove Associations
    - Deleting Generic Object Definitions
- Generic Objects
    - Querying Generic Objects
        - Request:
        - Response:
    - Creating Generic Objects
    - Editing Generic Objects
    - Calling Generic Object Methods
    - Deleting Generic Objects

Generic Object Management

Management of Generic Objects is provided with the following collections:

```
/api/generic_object_definitions
/api/generic_objects
```


- Managing Generic Object Definitions
    - Querying Generic Object Definitions
    - Creating Generic Object Definitions
    - Editing Generic Object Definitions
    - Deleting Generic Object Definitions
- Managing Generic Objects
    - Querying Generic Objects
    - Creating Generic Objects
    - Editing Generic Objects
    - Calling Generic Object Methods
    - Deleting Generic Objects

Generic Object Definitions

Querying Generic Object Definitions

Querying all available Generic Object Definitions:

```
GET /api/generic_object_definitions
```

Getting details on a specific Generic Object Definition:

```
GET /api/generic_object_definitions/:id
```

Getting details on a specific Generic Object Definition by name:

```
GET /api/generic_object_definitions/:name
```

Querying all the Generic Objects defined for a specific Generic Object Definition:

```
GET /api/generic_object_definitions/:id/generic_objects
```

Getting a specific Generic Object defined for a Generic Object Definition:

```
GET /api/generic_object_definitions/:id/generic_objects/:generic_object_id
```

Fetching the vms associations for a specific Generic Object:

```
GET /api/generic_object_definitions/:id/generic_objects/:generic_object_id?associations=vms
```

Creating Generic Object Definitions

Generic Object Definitions can be created as follows:

```
POST /api/generic_object_definitions
```

```
{
  "name"        : "LoadBalancer",
  "description" : "LoadBalancer description",
  "properties"  : {
    "attributes"  : {
      "address"      : "string",
      "last_restart" : "datetime"
    },
    "associations" : {
      "vms"      : "Vm",
      "services" : "Service"
    },
    "methods" : [
      "add_vm",
      "remove_vm"
    ]
  },
  "picture"     : {
    "extension" : "png",
    "content"   : "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAYAAAAfFcSJAAAABGdBTUEAALGP"\
      "C/xhBQAAACBjSFJNAAB6JgAAgIQAAPoAAACA6AAAdTAAAOpgAAA6mAAAF3Cc"\
      "ulE8AAAACXBIWXMAAAsTAAALEwEAmpwYAAABWWlUWHRYTUw6Y29tLmFkb2Jl"\
      "LnhtcAAAAAAAPHg6eG1wbWV0YSB4bWxuczp4PSJhZG9iZTpucptZXRhLyIg"\
      "eDp4bXB0az0iWE1QIENvcmUgNS40LjAiPgogICA8cmRmOlJERiB4bWxuczpy"\
      "ZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1u"\
      "cyMiPgogICAgICA8cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0iIgogICAg"\
      "ICAgICB4bWxuczp0aWZmPSJodHRwOi8vbnMuYWRvYmUuY29tL3RpZmYv"\
      "MS4wLyI+CiAgICAgICAgICAgIDx0aWZmOk9yaWVudGF0aW9uPjE8L3RpZmY6T3Jp"\
      "ZW50YXRpb24+CiAgICAgICAgICAgIDwvcmRmOkRlc2NyaXB0aW9uPgogICA8L3JkZjpS"\
      "REY+CjwveDp4bXBtZXRhPgpMwidZAAAADUlEQVQIHWNgYGCwBQAAQgA+3N0+"\
      "xQAAAABJRU5ErkJggg=="
  }
}
```

Example of creating a Generic Object Definition with a picture reference:

```
{
  "name"        : "LoadBalancer",
  "description" : "LoadBalancer description",
  "properties"  : {
    "attributes"  : {
      "address"      : "string",
      "last_restart" : "datetime"
    },
    "associations" : {
      "vms"      : "Vm",
      "services" : "Service"
    },
    "methods" : [
      "add_vm",
      "remove_vm"
    ]
  },
  "picture"     : {
    "href" : "http://localhost:3000/api/pictures/11"
  }
}
```

Note: picture references can be by *href* or *id*.

Editing Generic Object Definitions

Updating Generic Object Definitions can be done via the *edit* action as follows:

```
POST /api/generic_object_definitions/:id
```

or by name:

```
POST /api/generic_object_definitions/:name
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "Latest LoadBalancer Description"
  }
}
```

Generic object definitions can also be updated with the PUT method as follows:

```
PUT /api/generic_object_definitions/:id
```

```
{
  "name" : "New LoadBalancer Name",
  "description" : "New LoadBalancer Description"
}
```

Pictures can be removed from Generic Object Definitions by sending in an empty picture reference via any of the edit methods, for example:

```
POST /api/generic_object_definitions/:name
```

```
{
  "action" : "edit",
  "resource" : {
    "picture" : {}
  }
}
```

Generic Object Definitions can be edited by href, id or name in bulk as in the following example:

```
POST /api/generic_object_definitions
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "name" : "Load Balancer",
      "resource" : {
        "name" : "Enhanced Load Balancer",
        "description" : "Description for Enhanced Load Balancer",
        "picture" : { "id" : "16" }
      }
    },
    {
      "href" : "http://localhost:3000/api/generic_object_definitions/12",
      "resource" : {
        "name" : "NetworkScanner",
        "description" : "Description for Network Scanner"
      }
    },
    {
      "id" : "13",
      "resource" : {
        "name" : "NetworkFirewall",
        "description" : "Description for Network Firewall"
      }
    }
  ]
}
```

Editing Generic Object Definitions can also be done by using one of the following actions for updating *attributes*, *methods* and *associations*:

- add_attributes
- remove_attributes
- add_methods
- remove_methods
- add_associations
- remove_associations

Add Attributes

```
POST /api/generic_object_definitions/:id
```

```
{
  "action" : "add_attributes",
  "resource" : {
    "attributes" : {
      "optional_attribute1" : "string",
      "optional_attribute2" : "boolean"
    }
  }
}
```

Remove Attributes

```
POST /api/generic_object_definitions/:id

{
  "action" : "remove_attributes",
  "resource" : {
    "attributes" : {
      "unused_attribute9" : "string"
    }
  }
}
```

Add Methods

```
POST /api/generic_object_definitions/:id

{
  "action" : "add_methods",
  "resource" : {
    "methods" : [ "go_method1", "go_method2" ]
  }
}
```

Remove Methods

```
POST /api/generic_object_definitions/:id

{
  "action" : "remove_methods",
  "resource" : {
    "methods" : [ "go_method3" ]
  }
}
```

Add Associations

```
POST /api/generic_object_definitions/:id

{
  "action" : "add_associations",
  "resource" : {
    "associations" : {
      "az" : "AvailabilityZone",
      "chargeback" : "ChargebackVm"
    }
  }
}
```

Remove Associations

```
POST /api/generic_object_definitions/:id

{
  "action" : "remove_associations",
  "resource" : {
    "associations" : {
      "chargeback" : "ChargebackVm"
    }
  }
}
```

Deleting Generic Object Definitions

Generic Object Definitions can be deleted via the *delete* action as follows:

```
POST /api/generic_object_definitions/:id
```

or by name:

```
POST /api/generic_object_definitions/:name

{
  "action" : "delete"
}
```

Multiple Generic Object Definitions can also be deleted in a single request as follows:

```
POST /api/generic_object_definitions

{
  "action" : "delete",
```

```
  "resources" : [
    { "href" : "http://localhost:3000/api/generic_object_definitions/11" },
    { "id" : "12" }
  ]
}
```

Single Generic Object Definitions can also be deleted with the DELETE method as follows:

```
DELETE /api/generic_object_definitions/:id
```

or by name:

```
DELETE /api/generic_object_definitions/:name
```

Generic Objects

Querying Generic Objects

Querying all available Generic Objects:

```
GET /api/generic_objects
```

Getting details on a specific Generic Object:

```
GET /api/generic_objects/:id
```

Querying all the Generic Objects defined for a specific Service:

```
GET /api/services/:id/generic_objects
```

Querying Generic Objects and related picture and image_href for a specific Service:

Request:

```
GET /api/services/:id?expand=generic_objects&attributes=generic_objects.picture
```

Response:

```
{
  "href" : "http://localhost:3000/api/services/645",
  "id" : "645",
  "name" : "Test Azure Service",
  "description" : "Azure",
  "guid" : "6ce8f6c2-7f93-4a7e-b618-da9acab470ac",
  "options": {  },
  "display": false,
  "created_at" : "2017-08-07T16:52:46Z",
  "updated_at" : "2017-08-07T16:52:46Z",
  "miq_group_id" : "1",
  "retired": false,
  "tenant_id" : "1",
  "initiator" : "user",
  "generic_objects": [
    {
      "href" : "http://localhost:3000/api/services/645/generic_objects/1",
      "id" : "1",
      "name" : "generic_object_1",
      "generic_object_definition_id" : "1",
      "created_at" : "2017-08-15T17:35:08Z",
      "updated_at" : "2017-09-07T12:05:14Z",
      "properties": {
         "vms" : [ 615 ],
         "start" : true
      },
      "picture" : {
        "href" : "http://localhost:3000/api/picture/1",
        "id" : "1",
        "resource_id" : "1",
        "resource_type" : "GenericObjectDefinition",
        "image_href" : "http://localhost:3000/pictures/1.png",
        "extension" : "png"
      }
    }
  ],
  "actions" : {
    ...
  }
}
```

Creating Generic Objects

Generic Objects can be created as follows:

```
POST /api/generic_objects
```

```
{
  "generic_object_definition" : { "href" : "http://localhost:3000/api/generic_object_definitions/11" },
  "name" : "go1_name",
  "uid" : "optional_uid",
  "property_attributes" : {
    "widget" : "widget value",
    "supports_feature_a" : false
  },
  "associations" : {
    "vms" : [
      { "href" : "http://localhost:3000/api/vms/11" },
      { "id" : "12 }
    ]
  }
}
```

Generic Objects can also be directly created for a specific Generic Object Definition type as follows:

```
POST /api/generic_object_definitions/:id/generic_objects
```

```
{
  "name" : "go1_name",
  "uid" : "optional_uid",
  "property_attributes" : {
    "widget" : "widget value",
    "supports_feature_a" : false
  },
  "associations" : {
    "vms" : [
      { "href" : "http://localhost:3000/api/vms/11" },
      { "id" : "12 }
    ]
  }
}
```

In which case, the generic_object_definition reference is not needed.

Editing Generic Objects

Updating Generic Objects can be done via the *edit* action as follows:

```
POST /api/generic_objects/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "name" : "updated_name",
    "property_attributes" : {
      "widget" : "updated_widget"
    }
  }
}
```

Calling Generic Object Methods

When generic objects have methods defined in their definitions, those methods are exposed with the generic objects as actions.

The following example shows a generic object with two methods *method_a* and *method_b*:

```
GET /api/generic_objects/101
```

```
{
  "href" : "http://localhost:3000/api/generic_objects/101",
  "id"   : "101",
  "name" : "sample_object",
  "generic_object_definition_id" : "5",
  "property_attributes" : {
    "widget" : "a sample widget",
    "object_flag" : true
  },
  "actions" : [
    {
      "name" : "edit",
      "method" : "post",
      "href" : "http://localhost:3000/api/generic_objects/101"
```

```
    },
    {
      "name" : "delete",
      "method" : "post",
      "href" : "http://localhost:3000/api/generic_objects/101"
    },
    {
      "name" : "delete",
      "method" : "delete",
      "href" : "http://localhost:3000/api/generic_objects/101"
    },
    {
      "name" : "method_a",
      "method" : "post",
      "href" : "http://localhost:3000/api/generic_objects/101"
    },
    {
      "name" : "method_b",
      "method" : "post",
      "href" : "http://localhost:3000/api/generic_objects/101"
    }
  ]
}
```

With the above sample generic object, the custom methods can be invoked like any other action as follows:

```
POST /api/generic_objects/101
```

```
{
  "action" : "method_a",
  "resource" : {
    "attr1" : "value 1",
    "attr2" : "value 2"
  }
}
```

Deleting Generic Objects

Generic Objects can be deleted via the *delete* action as follows:

```
POST /api/generic_objects/:id
```

```
{
  "action" : "delete"
}
```

Multiple Generic Objects can also be deleted in a single request as follows:

```
POST /api/generic_objects
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/generic_objects/21" },
    { "id" : "22" },
    { "id" : "23" }
  ]
}
```

Single Generic Objects can also be deleted with the DELETE method as follows:

```
DELETE /api/generic_objects/:id
```


- Group Management
  - Querying Groups
  - Creating Groups
  - Editing Groups
  - Deleting Groups

Group Management

Management of User Groups is provided via the following collection:

```
/api/groups
```

Full CRUD actions on groups is available:

- Querying Groups

- Creating Groups
- Editing Groups
- Deleting Groups

Querying Groups

Querying all groups in the system:

```
GET /api/groups
```

Getting details on a specific group:

```
GET /api/groups/:id
```

Creating Groups

Groups can be created via a POST to the groups collection or via the **create** action signature which also allows creation of multiple groups in a single request.

```
POST /api/groups
```

```
{
  "description" : "sample_group",
  "role" : { "id" : "2" },
  "tenant" : { "href" : "http://localhost:3000/api/tenants/1" },
  "filters" : {
    "belongsto" : [ "/managed/area/1", "/managed/area/2" ],
    "managed" : [[ "/managed/infra/1", "/managed/infra/2" ]]
  }
}
```

Note:

role can be specified via **name**, **href**, or **id**

tenant can be specified via **href** or **id**

or creating multiple groups:

```
{
  "action" : "create",
  "resources" : [
    { "description" : "sample_group1", ... },
    { "description" : "sample_group2", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Groups.

Editing Groups

```
POST /api/groups/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "updated_sample_group",
    "filters" : {
      "belongsto" : [ "/managed/area/1", "/managed/area/2", "/managed/area/3" ],
      "managed" : [[ "/managed/infra/1", "/managed/infra/2"], ["/managed/other/3"]]
    }
  }
}
```

or editing multiple groups:

```
POST /api/groups
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/groups/101",
      "description" : "updated_sample_group1"
    },
```

```
    {
      "href" : "http://localhost:3000/api/groups/102",
      "description" : "updated_sample_group2"
    },
    ...
  ]
}
```

Deleting Groups

groups can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/groups/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/groups/101
```

Deleting multiple groups can be done as follows:

```
POST /api/groups
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/groups/101" },
    { "href" : "http://localhost:3000/api/groups/102" },
    ...
  ]
}
```


- Host Management
  - Querying Hosts
  - Editing a Host

Host Management

Management of Hosts adds support of the following actions:

Action	Description
edit	Edit a Host

- Querying Hosts
- Editing a Host

Querying Hosts

Hosts are queried via the primary collection URL:

```
/api/hosts
```

Editing a Host

Currently, only the host password can be edited via the API. All other edit requests will be rejected.

The host password is provided by a *credentials* hash similar to the signature supported for providers.

```
POST /api/hosts/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "credentials" : {
      "password" : "updated_password"
    }
  }
}
```

When not specified, the authentication type used for the password is *default*:

```
POST /api/hosts/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "credentials" : {
      "auth_type" : "default",
      "password" : "updated_password"
    }
  }
}
```

Password for multiple hosts can also be updated with a single request as follows:

```
POST /api/hosts
```

```
{
  "action" : "edit",
  "resources" : [
    { "id" : "101", "credentials" : { "password" : "updated_password" } },
    { "id" : "102", "credentials" : { "password" : "updated_password" } },
    { "id" : "103", "credentials" : { "password" : "updated_password" } },
    ...
  ]
}
```

Note:

host can be referenced by **href** or **id**

- Instance Management
  - Targeting Instances
  - Querying Instances
    - Load Balancers
  - Starting an Instance
  - Stopping an Instance
  - Suspending an Instance
  - Pausing an Instance
  - Terminating an Instance
  - Shelving an Instance
  - Rebooting Guest in an Instance
  - Resetting an Instance

Instance Management

Management of instances adds support for the following actions:

Action	Description
start	Starts an Instance
stop	Stops an Instance
suspend	Suspending an Instance
pause	Pausing an Instance
reboot_guest	Rebooting Guest in an Instance
reset	Resetting an Instance
shelve	Shelving an Instance
terminate	Terminating an Instance

- Targeting Instances
- Querying Instances
- Starting an Instance
- Stopping an Instance
- Suspending an Instance
- Pausing an Instance
- Rebooting Guest in an Instance
- Resetting an Instance
- Shelving an Instance

- Terminating an Instance

Targeting Instances

These actions can be triggered on individual instance resources:

```
/api/instances/:id
```

As simply as POSTing the following action to an instance.

```
{
  "action" : "start"
}
```

Requests can also be made on multiple instances by targeting the primary collection:

```
/api/instances
```

```
{
  "action" : "start",
  "resources" : [
    { "href" : "http://localhost:3000/api/instances/101" },
    { "href" : "http://localhost:3000/api/instances/102" },
    ...
  ]
}
```

Querying Instances

Instances are queried via the primary collection URL:

```
/api/instances
```

Filtering, sorting and paging as mentioned on the Querying page.

Load Balancers

Load Balancers of an instance can be queried via the load_balancers subcollection as follows:

Query all Load Balancers of an instance:

```
GET /api/instances/:id/load_balancers
```

Query a specific Load Balancer of an instance:

```
GET /api/instances/:id/load_balancers/:load_balancer_id
```

Or expanding the Load Balancers for the specific instance as follows:

```
GET /api/instances/:id?expand=load_balancers
```

Starting an Instance

```
{
  "action" : "start"
}
```

Stopping an Instance

```
{
  "action" : "stop"
}
```

Suspending an Instance

```
{
  "action" : "suspend"
}
```

Pausing an Instance

```
{
  "action" : "pause"
}
```

Terminating an Instance

```
{
  "action" : "terminate"
```

```
}
```

Shelving an Instance

```
{
  "action" : "shelve"
}
```

Rebooting Guest in an Instance

```
{
  "action" : "reboot-guest"
}
```

Resetting an Instance

```
{
  "action" : "reset"
}
```


- Metric Rollups
  - Getting 60 Daily Metric Rollups
  - Getting Hourly Metric Rollups for a range of Dates
  - Getting Hourly Metric Rollups for a Service
  - Getting 30 Daily Metric Rollups for a Vm

Metric Rollups

Querying of Metric Rollups is made available via the following collection:

```
/api/metric_rollups
```

Supported query parameters:

Parameter	Examples
resource_type	VmOrTemplate, Service or any resource type that supports Metric Rollups
resource_ids[]	id1
resource_ids[]	id2
resource_ids[]	...
capture_interval	hourly, daily
start_date	2018-01-10, 2018-01-10T11:20:00Z
end_date	2018-01-17, 2017-12-31T23:59:00Z

Getting 60 Daily Metric Rollups

```
GET /api/metric_rollups?resource_type=VmOrTemplate
                       &resource_ids[]=101
                       &resource_ids[]=102
                       &resource_ids[]=103
                       &capture_interval=daily
                       &start_date='2017-10-01'
                       &limit=60
```

Getting Hourly Metric Rollups for a range of Dates

```
GET /api/metric_rollups?resource_type=Service
                       &resource_ids[]=55
                       &resource_ids[]=56
                       &capture_interval=hourly
                       &start_date='2017-11-15'
                       &end_date='2017-11-18'
```

Metric rollups can also be obtained for a specific resource as a subcollection:

```
/api/vms/:id/metric_rollups
```

```
/api/services/:id/metric_rollups
```

Getting Hourly Metric Rollups for a Service

```
GET /api/services/57/metric_rollups?capture_interval=hourly&start_date='2017-06-15'
```

Getting 30 Daily Metric Rollups for a Vm

```
GET /api/vms/104/metric_rollups?capture_interval=daily&start_date='2017-01-01'&limit=30
```


- Network Routers
  - Querying Network Routers
  - Deleting Network Routers
    - Request:
    - Response:

Network Routers

Management of Network Routers is provided via the following collection:

```
/api/network_routers
```

The following actions are available on Network Routers:

- Querying Network Routers
- Deleting Network Routers

Querying Network Routers

Querying all Network Routers in the system is simply:

```
GET /api/network_routers
```

Getting details on a specific Network Router:

```
GET /api/network_routers/:id
```

Deleting Network Routers

Network Routers can be deleted via either the **delete** POST action or via the DELETE HTTP method.

Request:

```
POST /api/network_routers/101

{
  "action" : "delete"
}
```

Response:

```
{
  "success" : true,
  "message" : "Deleting Networking Router id:101 name: 'test_network_router'",
  "task_id" : "33",
  "task_href" : "http://localhost:3000/api/tasks/33"
}
```

or simply:

```
DELETE /api/network_routers/101
```

Deleting multiple Network Routers can be done as follows:

```
POST /api/network_routers

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/network_routers/101" },
    { "href" : "http://localhost:3000/api/network_routers/102" },
    ...
  ]
}
```


- Notifications Support
- Querying Notifications
- Marking Notifications as Seen
  - Deleting Notifications

Notifications Support

Support for Notifications to the authenticated user is provided via the following collection:

`/api/notifications`

- Querying Notifications
- Marking Notifications as Seen
- Deleting Notifications

Querying Notifications

The following fetches the current notifications for the authenticated user:

```
GET /api/notifications
```

```
{
  "name": "notifications",
  "count": 3,
  "subcount": 3,
  "resources": [
    {
      "href": "http://localhost:3000/api/notifications/44"
    },
    {
      "href": "http://localhost:3000/api/notifications/45"
    },
    {
      "href": "http://localhost:3000/api/notifications/46"
    }
  ],
  "actions": [
    {
      "name": "mark_as_seen",
      "method": "post",
      "href": "http://localhost:3000/api/notifications"
    },
    {
      "name": "delete",
      "method": "post",
      "href": "http://localhost:3000/api/notifications"
    }
  ]
}
```

Or fetching additional details on the notifications as follows:

```
GET /api/notifications?expand=resources&attributes=details
```

```
{
  "name": "notifications",
  "count": 3,
  "subcount": 3,
  "resources": [
    {
      "href": "http://localhost:3000/api/notifications/44",
      "id": "44",
      "user_id": "1",
      "seen": false,
      "details": {
        "level": "info",
        "created_at": "2016-11-03T17:23:29Z",
        "text": "Vm [aab-test] was powered on",
        "bindings": {
          "initiator": {
            "link": {
              "id": "1",
              "model": "User"
            },
            "text": "Administrator"
          }
        },
        "seen": false
      },
      "actions": [
        {
          "name": "mark_as_seen",
          "method": "post",
          "href": "http://localhost:3000/api/notifications/44"
        },
```

```
        {
          "name": "delete",
          "method": "post",
          "href": "http://localhost:3000/api/notifications/44"
        },
        {
          "name": "delete",
          "method": "delete",
          "href": "http://localhost:3000/api/notifications/44"
        }
      ]
    },
    {
      "href": "http://localhost:3000/api/notifications/45",
      "id": "45",
      "notification_id": "22",
      "user_id": "1",
      "seen": true,
      "details": {
        "level": "info",
        "created_at": "2016-11-03T17:23:29Z",
        "text": "Vm [aab-test2] was powered on",
        "bindings": {
          "initiator": {
            "link": {
              "id": "1",
              "model": "User"
            },
            "text": "Administrator"
          }
        },
        "seen": true
      },
      "actions": [
        {
          "name": "mark_as_seen",
          "method": "post",
          "href": "http://localhost:3000/api/notifications/45"
        },
        {
          "name": "delete",
          "method": "post",
          "href": "http://localhost:3000/api/notifications/45"
        },
        {
          "name": "delete",
          "method": "delete",
          "href": "http://localhost:3000/api/notifications/45"
        }
      ]
    },
    {
      "href": "http://localhost:3000/api/notifications/46",
      "id": "46",
      "notification_id": "23",
      "user_id": "1",
      "seen": false,
      "details": {
        "level": "info",
        "created_at": "2016-11-03T17:23:54Z",
        "text": "Vm [aab-dev] was powered off",
        "bindings": {
          "initiator": {
            "link": {
              "id": "1",
              "model": "User"
            },
            "text": "Administrator"
          }
        },
        "seen": false
      },
      "actions": [
        {
          "name": "mark_as_seen",
          "method": "post",
          "href": "http://localhost:3000/api/notifications/46"
        },
        {
          "name": "delete",
          "method": "post",
```

```
          "href": "http://localhost:3000/api/notifications/46"
        },
        {
          "name": "delete",
          "method": "delete",
          "href": "http://localhost:3000/api/notifications/46"
        }
      ]
    }
  ],
  "actions": [
    {
      "name": "mark_as_seen",
      "method": "post",
      "href": "http://localhost:3000/api/notifications"
    },
    {
      "name": "delete",
      "method": "post",
      "href": "http://localhost:3000/api/notifications"
    }
  ]
}
```

Marking Notifications as Seen

Marking Notifications as seen for the authenticated user is done via the **mark_as_seen** action as follows:

```
POST /api/notifications/:id

{
  "action" : "mark_as_seen"
}
```

Or, marking several notifications as seen in a single call as follows:

```
POST /api/notifications

{
  "action" : "mark_as_seen",
  "resources" : [
    { "href" : "http://localhost:3000/api/notifications/44" },
    { "href" : "http://localhost:3000/api/notifications/45" },
    ...
  ]
}
```

Deleting Notifications

Deleting a notification is available via the **delete** action or DELETE on the individual notification as follows:

```
POST /api/notifications/:id

{
  "action" : "delete"
}
```

Or simply:

```
DELETE /api/notifications/:id
```

Also, deleting multiple notifications is available via the **delete** action on the collection passing in references to the different notifications:

```
POST /api/notifications

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/notifications/44" },
    { "href" : "http://localhost:3000/api/notifications/45" },
    ...
  ]
}
```


- Orchestration Support
  - Orchestration Templates ===
  - Querying Orchestration Templates
  - Creating Orchestration Templates
  - Editing Orchestration Templates

- Copying Orchestration Templates
- Deleting Orchestration Templates

Orchestration Support

Support for orchestration is provided by the following:

- Orchestration Templates

Orchestration Templates ===

Management of orchestration templates is provided with the following collection:

```
/api/orchestration_templates
```

Full CRUD actions on orchestration templates is available:

- Querying Orchestration Templates
- Creating Orchestration Templates
- Editing Orchestration Templates
- Copying Orchestration Templates
- Deleting Orchestration Templates

Querying Orchestration Templates

Querying all orchestration templates in the system:

```
GET /api/orchestration_templates
```

Getting details on a specific orchestration template:

```
GET /api/orchestration_templates/:id
```

Creating Orchestration Templates

Orchestration templates can be created via a POST to the orchestration templates collection or via the **create** action signature which also allows creation of multiple orchestration templates in a single request.

```
POST /api/orchestration_templates
```

```
{
  "name" : "azure-single-vm-from-user-image",
  "type" : "OrchestrationTemplateAzure",
  "description" : "Create a single VM from a user image",
  "content" : " ... ",
  "orderable" : true,
  ...
}
```

or creating multiple orchestration templates:

```
{
  "action" : "create",
  "resources" : [
    { "description" : "sample_orchestration_template1", ... },
    { "description" : "sample_orchestration_template2", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating orchestration templates.

Editing Orchestration Templates

```
POST /api/orchestration_templates/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "updated_sample_orchestration_template",
  }
}
```

or editing multiple orchestration templates:

```
POST /api/orchestration_templates
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/orchestration_templates/101",
      "description" : "updated_sample_orchestration_template1"
    },
    {
      "href" : "http://localhost:3000/api/orchestration_templates/102",
      "description" : "updated_sample_orchestration_template2"
    },
    ...
  ]
}
```

Copying Orchestration Templates

This capability allows orchestration templates to be copied onto new orchestration templates with updated content via the *copy* action as follows:

```
POST /api/orchestration_templates/:id
```

```
{
  "action" : "copy",
  "resource" : {
    "content" : "{ 'Description' : 'Copied Test Content' }"
  }
}
```

or copying multiple orchestration templates:

```
POST /api/orchestration_templates
```

```
{
  "action" : "copy",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/orchestration_templates/101",
      "content" : "{ 'Description' : 'Copied Test Content 1' }"
    },
    {
      "href" : "http://localhost:3000/api/orchestration_templates/102",
      "content" : "{ 'Description' : 'Copied Test Content 2' }"
    },
    ...
  ]
}
```

Deleting Orchestration Templates

Orchestration templates can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/orchestration_templates/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/orchestration_templates/101
```

Deleting multiple orchestration templates can be done as follows:

```
POST /api/orchestration_templates
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/orchestration_templates/101" },
    { "href" : "http://localhost:3000/api/orchestration_templates/102" },
    ...
  ]
}
```


- Physical Server Management
  - Querying Physical Servers
  - Managing Physical Servers

- Powering On a Physical Server
- Powering Off a Physical Server
- Powering Off Now a Physical Server
- Refresh a Physical Server
- Restart a Physical Server
- Restart Now a Physical Server
- Restart a Physical Server to System Setup
- Restart Management Controller
- Blink the Identify LED
- Turn On the Identify LED
- Turn Off the Identify LED

Physical Server Management

Management of Physical Servers is available via the following endpoint:

```
/api/physical_servers
```

Querying Physical Servers

Querying all physical servers in the system is simply:

```
GET /api/physical_servers
```

Getting details on a specific Physical Server:

```
GET /api/physical_servers/:id
```

Managing Physical Servers

The following actions are available on Physical Servers:

Action	Description
power_on	Power On
power_off	Power Off
power_off_now	Power Off Now
refresh_now	Refresh
restart	Restart
restart_now	Restart Now
restart_to_sys_setup	Restart to System Setup
restart_mgmt_controller	Restart Management Controller
blink_loc_led	Blink the Identify LED
turn_on_loc_led	Turn On the Identify LED
turn_off_loc_led	Turn Off the Identify LED

Powering On a Physical Server

```
{
  "action" : "power_on"
}
```

Powering Off a Physical Server

```
{
  "action" : "power_off"
}
```

Powering Off Now a Physical Server

```
{
  "action" : "power_off_now"
}
```

Refresh a Physical Server

```
{
  "action" : "refresh"
}
```

Restart a Physical Server

```
{
  "action" : "restart"
}
```

Restart Now a Physical Server

```
{
  "action" : "restart_now"
}
```

Restart a Physical Server to System Setup

```
{
  "action" : "restart_to_sys_setup"
}
```

Restart Management Controller

```
{
  "action" : "restart_mgmt_controller"
}
```

Blink the Identify LED

```
{
  "action" : "blink_loc_led"
}
```

Turn On the Identify LED

```
{
  "action" : "turn_on_loc_led"
}
```

Turn Off the Identify LED

```
{
  "action" : "turn_off_loc_led"
}
```


- Picture Management
  - Creating a Picture
  - Querying Pictures

Picture Management

Management of Pictures adds support of the following actions:

- Creating a Picture
- Querying Pictures

Creating a Picture

```
POST /api/pictures
```

```
{
  "extension": "<extension of image>",
  "content": "<base 64 encoded image here>"
}
```

Supported extensions include:

Extension
jpg
png
svg

Querying Pictures

Querying pictures like any other collection:

```
GET /api/pictures?expand=resources
```

```
{
  "name": "pictures",
  "count": 5,
  "subcount": 5,
  "resources": [
    {
      "href": "http://localhost:3000/api/pictures/1",
      "id": "1",
      "image_href" : "http://localhost:3000/pictures/1.jpg",
      "extension" : "jpg"
    },
    ...
    {
      "href": "http://localhost:3000/api/pictures/4",
      "id": "4",
      "image_href" : "http://localhost:3000/pictures/4.png",
      "extension" : "png",
      "resource_id": "1",
      "resource_type": "ServiceTemplate"
    },
    {
      "href": "http://localhost:3000/api/pictures/5",
      "id": "5",
      "image_href" : "http://localhost:3000/pictures/5.jpg",
      "extension" : "jpg"
    }
  ],
  "actions": [
    {
      "name": "create",
      "method": "post",
      "href": "http://localhost:3000/api/pictures"
    }
  ]
}
```

This query does not return the pictures themselves, simply metadata about the pictures.

Actually fetching the particular picture is done by getting the image by its image_href.

```
GET /api/pictures/4
```

```
{
  "href": "http://localhost:3000/api/pictures/4",
  "id": "4",
  "resource_id": "1",
  "resource_type": "ServiceTemplate",
  "image_href": "http://localhost:3000/pictures/4.jpg"
}
```

In which case the image can simply be downloaded with a GET:

```
GET http://localhost:3000/pictures/4.jpg
```

For fetching information about pictures pertaining to services, see the section Querying Pictures for details.

- Policy Management
  - Querying Policies
  - Querying Policies of a Resource
  - Creating Policies
  - Editing Policies
  - Deleting Policies
  - Policy and Policy Profiles Actions
  - Resource Policy Management
    - Policy specification
    - Assigning policies to a resource
      - Response:
    - Unassigning policies from a resource
      - Response:
  - Resolving Policies of a Resource
    - Response:
  - Resolving Policy Profiles of a Resource
    - Response:

Policy Management

Management of policies and policy profiles is provided in the following forms:

- Querying of policies via the primary collection:

`/api/policies`

- Querying of policy profiles via the primary collection:

`/api/policy_profiles`

- Assigning and Unassigning policies to resources via the policies subcollection of a resource

`/api/:collection/:id/policies`

and posting **assign** or **unassign** actions to the policies subcollection.

- Assigning and Unassigning policy profiles to resources via the policy_profiles subcollection of a resource

`/api/:collection/:id/policy_profiles`

and posting **assign** or **unassign** actions to the policy_profiles subcollection.

- Querying Policies
- Querying Policies of a Resource
- Creating Policies
- Editing Policies
- Deleting Policies
- Policy and Policy Profiles Actions
- Resource Policy Management
- Policy specification
- Assigning policies to a resource
- Unassigning policies from a resource
- Resolving Policies of a Resource
- Resolving Policy Profiles of a Resource

Querying Policies

Querying all policies in the system is simply:

```
GET /api/policies
```

```
{
  "name": "policies",
  "count": 20,
  "subcount": 20,
  "resources": [
    {
      "href": "http://localhost:3000/api/policies/7"
    },
    {
      "href": "http://localhost:3000/api/policies/8"
    },
    ...
  ]
}
```

And expanding the policies, providing additional details:

```
GET /api/policies?expand=resources
```

```
{
  "name": "policies",
  "count": 3,
  "subcount": 3,
  "resources": [
    {
      "href": "http://localhost:3000/api/policies/7",
      "id": "7",
      "name": "b1fd23bc-acc4-11e4-abc7-b8e85646e742",
```

```
      "description": "test_control_policy",
      "created_on": "2015-02-04T23:22:32Z",
      "updated_on": "2015-04-09T13:20:58Z",
      "towhat": "Vm",
      "guid": "b1fd23bc-acc4-11e4-abc7-b8e85646e742",
      "created_by": "admin",
      "updated_by": "admin",
      "active": true,
      "mode": "control"
    },
    {
      "href": "http://localhost:3000/api/policies/8",
      "id": "8",
      "name": "68bfaade-ad3d-11e4-8d4e-b8e85646e742",
      "description": "server_type_control_policy",
      "created_on": "2015-02-05T13:46:39Z",
      "updated_on": "2015-04-09T13:20:58Z",
      "towhat": "Vm",
      "guid": "68bfaade-ad3d-11e4-8d4e-b8e85646e742",
      "created_by": "admin",
      "updated_by": "admin",
      "active": true,
      "mode": "control"
    },
    ...
  ]
}
```

Querying Policies of a Resource

To find out which policies are assigned to a resource, the policies subcollection can be expanded for that particular resource as follows:

```
GET /api/vms/320?expand=policies
```

This will provide the resources as well as the related policies

```
{
  "href": "http://localhost:3000/api/vms/320",
  "id": "320",
  "vendor": "vmware",
  "name": "aab-vm1",
  "description": "this is a test",
  "raw_power_state": "poweredOn",
  ...
  "policies": [
    {
      "href": "http://localhost:3000/api/vms/320/policies/7",
      "id": "7",
      "name": "b1fd23bc-acc4-11e4-abc7-b8e85646e742",
      "description": "test_control_policy",
      "created_on": "2015-02-04T23:22:32Z",
      "updated_on": "2015-04-09T13:20:58Z",
      "towhat": "Vm",
      "guid": "b1fd23bc-acc4-11e4-abc7-b8e85646e742",
      "created_by": "admin",
      "updated_by": "admin",
      "active": true,
      "mode": "control"
    }
  ]
}
```

One can also simply query the policies of a resource by the subcollection as follows:

```
GET /api/vms/320/policies
```

```
{
  "name": "policies",
  "count": 3,
  "subcount": 1,
  "resources": [
    {
      "href": "http://localhost:3000/api/vms/320/policies/7"
    }
  ],
  "actions": [
    ...
  ]
}
```

Finding the policies that are part of a policy profile is queried the same as regular resources

```
GET /api/policy_profiles/:id?expand=policies
```

or just the policies themselves as follows:

```
GET /api/policy_profiles/:id/policies
```

Creating Policies

Policies can be created via a POST to the policies collection or via the **create** action signature which also allows creation of multiple policies in a single request.

```
POST /api/policies
```

```
{
  "name" : "sample_policy",
  "description" : "Sample Policy",
  "mode" : "compliance",
  "towhat" : "ManageIQ::Providers::Redhat::InfraManager",
  "condition_ids" : [11, 12],
  "policy_contents" : [
    {
      "event_id" : 201,
      "actions" : [
        {
          "action_id" : 3001,
          "opts" : { "qualifier" : "failure" }
        }
      ]
    }
  ]
}
```

or creating multiple policies:

```
{
  "action" : "create",
  "resources" : [
    { "name" : "sample_policy_1", "description" : "Sample Policy 1", ... },
    { "name" : "sample_policy_2", "description" : "Sample Policy 2", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating policies.

Editing Policies

```
POST /api/policies/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "Updated Policy Description"
  }
}
```

or editing multiple policies:

```
POST /api/policies
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/policies/101",
      "description" : "Updated Policy Description 1"
    },
    {
      "href" : "http://localhost:3000/api/policies/102",
      "description" : "Updated Policy Description 2"
    },
    ...
  ]
}
```

Deleting Policies

Policies can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/policies/101

{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/policies/101
```

Deleting multiple policies can be done as follows:

```
POST /api/policies

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/policies/101" },
    { "href" : "http://localhost:3000/api/policies/102" },
    ...
  ]
}
```

Policy and Policy Profiles Actions

For managing policies and policy profiles on resource three available actions are available. These are:

Action	Description
assign	Assign a policy or policy profile to the resource
unassign	Unassign a policy or policy profile from the resource
resolve	Resolves a resource policy or policy profile

Resource Policy Management

Policy management on resources can be done by POSTing **assign** and **unassign** actions to the policies or policy_profiles subcollection of resources. Policy management is available on the following primary collections:

Collection
/api/clusters
/api/hosts
/api/policy_profiles
/api/providers
/api/resource_pools
/api/templates
/api/vms

Policy specification

Policies and policy profiles can be specified using one of the following forms:

By href:

```
{
  "href" : "http://localhost:3000/api/policies/:id"
}

{
  "href" : "http://localhost:3000/api/policy_profiles/:id"
}
```

Or by policy Guid:

```
{
  "guid" : "b1fd23bc-acc4-11e4-abc7-b8e85646e742"
}
```

Assigning policies to a resource

Assigning policies or policy_profiles to resources is done by posting an **assign** action against the policies or policy_profiles subcollection of a resource.

```
POST /api/vms/320/policies

{
  "action" : "assign",
```

```
  "resource" : { "href" : "http://localhost:3000/api/policies/7" }
}
```

or multiple policies:

```
{
  "action" : "assign",
  "resource" : [
    { "href" : "http://localhost:3000/api/policies/7" },
    { "href" : "http://localhost:3000/api/policies/10" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Policy: id:'7' description:'test_control_policy' guid:'b1fd23bc-acc4-11e4-abc7-
b8e85646e742'",
      "href": "http://localhost:3000/api/vms/320",
      "policy_id": "7",
      "policy_href": "http://localhost:3000/api/policies/7"
    }
  ]
}
```

Unassigning policies from a resource

Unassign policies or policy_profiles to resources is done by posting an **unassign** action against the policies or policy_profiles subcollection of a resource.

```
POST /api/vms/320/policies
```

```
{
  "action" : "unassign",
  "resource" : { "href" : "http://localhost:3000/api/policies/7" }
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Unassigning Policy: id:'7' description:'test_control_policy' guid:'b1fd23bc-acc4-11e4-abc7-
b8e85646e742'",
      "href": "http://localhost:3000/api/vms/320",
      "policy_id": "7",
      "policy_href": "http://localhost:3000/api/policies/7"
    }
  ]
}
```

Resolving Policies of a Resource

Resolving a resource policy by targeting the subcollection resource directly as follows:

```
POST /api/vms/320/policies/7
```

```
{
  "action" : "resolve"
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Resolving Policy: id:'7' description:'test_control_policy' guid:'b1fd23bc-acc4-11e4-abc7-
b8e85646e742'",
      "result": [
        {
          "id": "7",
          "name": "b1fd23bc-acc4-11e4-abc7-b8e85646e742",
          "description": "test_control_policy",
          "created_on": "2015-02-04T23:22:32Z",
```

```
          "updated_on": "2015-04-09T13:20:58Z",
          "towhat": "Vm",
          "guid": "b1fd23bc-acc4-11e4-abc7-b8e85646e742",
          "created_by": "admin",
          "updated_by": "admin",
          "active": true,
          "mode": "control",
          "result": "allow",
          "conditions": [

          ],
          "actions": [

          ]
        }
      ],
      "href": "http://localhost:3000/api/vms/320",
      "policy_id": "7",
      "policy_href": "http://localhost:3000/api/policies/7"
    }
  ]
}
```

Resolving Policy Profiles of a Resource

Resolving a resource policy profile by targeting the subcollection resource directly as follows:

```
POST /api/vms/320/policy_profiles/34

{
  "action" : "resolve"
}
```

Response:

```
{
  "success": true,
  "message": "Resolving Policy Profile: id:'34' description:'Control Policies' guid:'f39b25e2-ad3e-11e4-8d4e-
b8e85646e742'",
  "result": [
    {
      "id": "34",
      "name": "f39b25e2-ad3e-11e4-8d4e-b8e85646e742",
      "description": "Control Policies",
      "set_type": "MiqPolicySet",
      "created_on": "2015-02-05T13:57:41Z",
      "updated_on": "2015-02-26T13:42:43Z",
      "guid": "f39b25e2-ad3e-11e4-8d4e-b8e85646e742",
      "mode": "control",
      "result": "allow",
      "policies": [
        {
          "id": "7",
          "name": "b1fd23bc-acc4-11e4-abc7-b8e85646e742",
          "description": "test_control_policy",
          "created_on": "2015-02-04T23:22:32Z",
          "updated_on": "2015-04-09T13:20:58Z",
          "towhat": "Vm",
          "guid": "b1fd23bc-acc4-11e4-abc7-b8e85646e742",
          "created_by": "admin",
          "updated_by": "admin",
          "active": true,
          "mode": "control",
          "result": "allow",
          "conditions": [

          ],
          "actions": [

          ]
        },
        {
          "id": "8",
          "name": "68bfaade-ad3d-11e4-8d4e-b8e85646e742",
          "description": "second_test_control_policy",
          "created_on": "2015-02-05T13:46:39Z",
          "updated_on": "2015-04-09T13:20:58Z",
          "towhat": "Vm",
          "guid": "68bfaade-ad3d-11e4-8d4e-b8e85646e742",
          "created_by": "admin",
          "updated_by": "admin",
```

```
          "active": true,
          "mode": "control",
          "result": "allow",
          "conditions": [

          ],
          "actions": [

          ]
        }
      ]
    }
  ],
  "href": "http://localhost:3000/api/vms/320",
  "policy_profile_id": "34",
  "policy_profile_href": "http://localhost:3000/api/policy_profiles/34"
}
```


- Policy Collections
- Policy Actions
  - Response:
- Policy Conditions
  - Response:
- Policy Events
  - Response:

Policy Collections

Primary Policy and Policy Profile collections:

`/api/policies`

`/api/policy_profiles`

With Policy Profiles being sets of Policies with the ability to query policies of a specific policy profile via:

`GET /api/policy_profiles/:id?expand=policies`

Or just the subcollection:

`GET /api/policy_profiles/:id/policies`

- Policy Actions
- Policy Conditions
- Policy Events

Policy Actions

Policy actions can be queried as a primary collection or as a subcollection of policies as follows:

`GET /api/policy_actions`

`GET /api/policies/:id?expand=policy_actions`

or simply the subcollection:

`GET /api/policies/:id/policy_actions`

In this example, we're expanding all policy actions

`GET /api/policy_actions?expand=resources`

Response:

```
{
  "name": "policy_actions",
  "count": 33,
  "subcount": 33,
  "resources": [
    {
      "href": "http://localhost:3000/api/policy_actions/1",
      "id": "1",
      "name": "audit",
      "description": "Generate Audit Event",
```

```
      "created_on": "2014-11-20T19:10:05Z",
      "updated_on": "2014-11-20T19:10:05Z",
      "guid": "d619567a-70e8-11e4-a5b4-b8e85646e742",
      "action_type": "default",
      "options": {
      }
    },
    {
      "href": "http://localhost:3000/api/policy_actions/2",
      "id": "2",
      "name": "log",
      "description": "Generate log message",
      "created_on": "2014-11-20T19:10:05Z",
      "updated_on": "2014-11-20T19:10:05Z",
      "guid": "d61c3be2-70e8-11e4-a5b4-b8e85646e742",
      "action_type": "default",
      "options": {
      }
    },
    {
      "href": "http://localhost:3000/api/policy_actions/3",
      "id": "3",
      "name": "evm_event",
      "description": "Show EVM Event on Timeline",
      "created_on": "2014-11-20T19:10:05Z",
      "updated_on": "2014-11-20T19:10:05Z",
      "guid": "d61cb05e-70e8-11e4-a5b4-b8e85646e742",
      "action_type": "default",
      "options": {
      }
    },
    {
      "href": "http://localhost:3000/api/policy_actions/4",
      "id": "4",
      "name": "script",
      "description": "Execute an external script",
      "created_on": "2014-11-20T19:10:05Z",
      "updated_on": "2014-11-20T19:10:05Z",
      "guid": "d61d225a-70e8-11e4-a5b4-b8e85646e742",
      "action_type": "default",
      "options": {
      }
    },
    ...
  ]
}
```

Policy Conditions

Policy conditions can be queried as a primary collection or as a subcollection of policies as follows:

```
GET /api/conditions
```

```
GET /api/policies/:id?expand=conditions
```

or simply the subcollection:

```
GET /api/policies/:id/conditions
```

In this example, we're expanding the conditions of a policy:

```
GET /api/policies/10?expand=conditions
```

Response:

```
{
  "href": "http://localhost:3000/api/policies/10",
  "id": "10",
  "name": "85dd1f68-ae53-11e4-9c45-b8e85646e742",
  "description": "web_server_policy",
  "created_on": "2015-02-06T22:57:27Z",
  "updated_on": "2015-04-09T13:20:58Z",
  "towhat": "Vm",
  "guid": "85dd1f68-ae53-11e4-9c45-b8e85646e742",
  "created_by": "admin",
  "updated_by": "admin",
  "active": true,
  "mode": "control",
  "conditions": [
    {
```

```
      "href": "http://localhost:3000/api/policies/10/conditions/1",
      "id": "1",
      "name": "fe75d0bc-860f-11e4-b12d-b8e85646e742",
      "description": "Memory For Web Server",
      "modifier": "allow",
      "expression": {
        "exp": {
          ">=": {
            "field": "Vm-num_cpu",
            "value": "2"
          }
        },
        "context_type": null
      },
      "towhat": "Vm",
      "created_on": "2014-12-17T17:13:17Z",
      "updated_on": "2015-02-08T15:16:16Z",
      "guid": "fe75d0bc-860f-11e4-b12d-b8e85646e742"
    }
  ]
}
```


Policy Events

Policy events can be queried as a primary collection or as a subcollection of policies as follows:

```
GET /api/events
```

```
GET /api/policies/:id?expand=events
```

or simply the subcollection:

```
GET /api/policies/:id/events
```

In this example, we're expanding all policy events

```
GET /api/events?expand=resources
```

Response:


```
{
  "name": "events",
  "count": 121,
  "subcount": 121,
  "resources": [
    {
      "href": "http://localhost:3000/api/events/1",
      "id": "1",
      "name": "evm_server_start",
      "description": "EVM Server Start",
      "created_on": "2014-11-20T19:10:11Z",
      "updated_on": "2014-11-20T19:10:11Z",
      "guid": "d9edd528-70e8-11e4-a5b4-b8e85646e742",
      "event_type": "Default"
    },
    {
      "href": "http://localhost:3000/api/events/2",
      "id": "2",
      "name": "evm_server_stop",
      "description": "EVM Server Stop",
      "created_on": "2014-11-20T19:10:11Z",
      "updated_on": "2014-11-20T19:10:11Z",
      "guid": "d9f44c00-70e8-11e4-a5b4-b8e85646e742",
      "event_type": "Default"
    },
    {
      "href": "http://localhost:3000/api/events/3",
      "id": "3",
      "name": "evm_server_not_responding",
      "description": "EVM Server Not Responding",
      "created_on": "2014-11-20T19:10:11Z",
      "updated_on": "2014-11-20T19:10:11Z",
      "guid": "d9f5dcaa-70e8-11e4-a5b4-b8e85646e742",
      "event_type": "Default"
    },
    ...
  ]
}
```


- Provider Support

- Querying Providers
  - Cloud Networks
  - Cloud Subnets
  - Load Balancers
  - Virtual Machines
- Creating Providers
- Specifying Credentials
- Specifying Connection Configurations
- Editing Providers
- Deleting Providers
- Refresh Providers
- Import Vm to Provider
- Foreman Support
- Provider Examples

Provider Support

Provider management including CRUD operations as well as the **refresh** action is available via the REST API. Management of providers is through:

`/api/providers`

- Querying Providers
- Creating Providers
- Specifying Credentials
- Specifying Connection Configurations
- Editing Providers
- Deleting Providers
- Refresh Providers
- Import Vm to Provider
- Foreman Support
- Provider Examples

Querying Providers

Providers can be queried as follows:

```
GET /api/providers
```

A specific provider can also be queried by its id:

```
GET /api/providers/:id
```

Foreman type providers can be queried by specifying the provider_class as follows:

```
GET /api/providers?provider_class=provider
```

Cloud Networks

The Cloud Networks of a provider can be queried by asking for the *cloud_networks* subcollection as follows:

```
GET /api/providers/:id/cloud_networks
```

```
GET /api/providers/:id/cloud_networks/:cloud_network_id
```

Or expanding the Cloud Networks in addition to the provider as follows:

```
GET /api/providers/:id?expand=cloud_networks
```

Cloud Subnets

The Cloud Subnets of a provider can be queried by asking for the *cloud_subnets* subcollection as follows:

```
GET /api/providers/:id/cloud_subnets
```

```
GET /api/providers/:id/cloud_subnets/:cloud_subnet_id
```

Or expanding the Cloud Subnets in addition to the provider as follows:

```
GET /api/providers/:id?expand=cloud_subnets
```

Load Balancers

Load Balancers of providers can be queried via the load_balancers subcollection as follows:

Query all Load Balancers of a provider:

```
GET /api/providers/:id/load_balancers
```

Query a specific Load Balancer of a provider:

```
GET /api/providers/:id/load_balancers/:load_balancer_id
```

Or expanding the Load Balancers for the specific provider as follows:

```
GET /api/providers/:id?expand=load_balancers
```

Virtual Machines

Virtual Machines of providers can be queried via the vms subcollection as follows:

Query all VMs of a provider:

```
GET /api/providers/:id/vms
```

Query a specific VM of a provider:

```
GET /api/providers/:id/vms/:vm_id
```

Or expanding the VMs for the specific provider as follows:

```
GET /api/providers/:id?expand=vms
```

Query specific VMs of a provider and get their name and vendor attributes:

```
GET /api/providers/:id/vms?filter[]=name='test*'&attributes=name,vendor&expand=resources
```

Creating Providers

Creating a provider is done via the **create** action as follows:

```
POST /api/providers
```

```
{
  "action" : "create",
  "resource" : {
    # New Provider JSON
  }
}
```

And like other **create** actions, the new provider data can simply be posted to the /api/providers URL.

```
POST /api/providers
```

```
{
  # New Provider JSON
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating providers.

For example, creating a VMware type provider:

```
POST /api/providers
```

```
{
  "name" : "vCenter 5",
  "type" : "ManageIQ::Providers::Vmware::InfraManager",
  "hostname" : "my_vcenter_50",
  "ipaddress" : "192.168.150.1",
  "zone" :  { "href" : "http://localhost:3000/api/zone/1" },
  "credentials" :  {
    "userid" : "vcenter_admin",
    "password" : "vcenter_password"
  }
}
```

If zone is not specified, the Default zone will be used.

The **type** attribute specifies the supported provider class names which include:

ManageIQ::Providers::Amazon::CloudManager
ManageIQ::Providers::Azure::CloudManager
ManageIQ::Providers::Kubernetes::ContainerManager
ManageIQ::Providers::Microsoft::InfraManager
ManageIQ::Providers::Openshift::ContainerManager
ManageIQ::Providers::Openstack::CloudManager
ManageIQ::Providers::Openstack::InfraManager
ManageIQ::Providers::Lenovo::PhysicalInfraManager
ManageIQ::Providers::Redhat::InfraManager
ManageIQ::Providers::Vmware::CloudManager
ManageIQ::Providers::Vmware::InfraManager

Note:

Please note that the provider **type** classes have changed in the appliance from earlier releases (*v2.0.0* of the API). Please refer to the Provider Types page for a mapping from the old to the new types.

Specifying Credentials

When creating or updating providers, the credentials can be specified as a a single default set, or a compound set where additional credentials are necessary, for example AMPQ for OpenStack or Metrics for RHEVM.

Single default credentials set:

```
{
  "name" : "vCenter 50",
  "type" : "ManageIQ::Providers::Vmware::InfraManager",
  ...
  "credentials" :  {
    "userid" : "vc_admin",
    "password" : "vc_password"
  }
}
```

Compound credentials set:

```
{
  "name" : "RHEVM",
  "type" : " ManageIQ::Providers::Redhat::InfraManager",
  ...
  "credentials" : [
     {
       "userid"    : "default_userid",
       "password"  : "default_password"
     },
     {
       "userid"    : "metrics_userid",
       "password"  : "metrics_password",
       "auth_type" : "metrics"
     }
  ]
}
```

Specifying Connection Configurations

When creating or updating providers, connection configurations can be set. Connection configurations can be used to specify resources such as an **amqp** event provider for OpenStack, adding Prometheus metrics, or an authentication token.

Prometheus metrics:

```
{
  "name": "Openshift Provider",
  "type": "ManageIQ::Providers::Openshift::ContainerManager",
  ...
  "connection_configurations": [
    {
      "endpoint": {
        "role"                  : "prometheus",
        "hostname"              : "prometheus_host",
        "port"                  : "443",
        "security_protocol"     : "ssl-without-validation",
```

```
        "certificate_authority" : "-----BEGIN CERTIFICATE-----",
        "verify_ssl": 0
      },
      "authentication": {
        "role"     :  "prometheus",
        "auth_key" :  "prometheus_auth_key"
      }
    }
  ]
}
```

Amqp event provider:

```
{
  "name": "Openstack Provider",
  "type": "ManageIQ::Providers::Openstack::CloudManager",
  ...
  "connection_configurations": [
    {
      "endpoint": {
        "role"              : "amqp",
        "hostname"          : "amqphost.com",
        "port"              : "5672",
        "security_protocol" : "non-ssl"
      },
      "authentication": {
        "userid"   : "amqp_userid",
        "password" : "amqp_password",
        "role"     : "amqp"
      }
    }
  ]
}
```

Ceilometer event provider:

```
{
  "name"              : "Openstack Provider",
  "type"              : "ManageIQ::Providers::Openstack::CloudManager",
  "hostname"          : "ceilometerhost.com",
  "ipaddress"         : "192.168.150.1",
  "security_protocol" : "ssl",
  ...
  "connection_configurations": {
      "userid"   :  "userid",
      "password" :  "password"
  },
  "connection_configurations": [
      "endpoint" : {
        "role"   : "ceilometer"
      }
    }
  ]
}
```

Editing Providers

Editing provider is available via the **edit** action with the **resource** specifying the attributes to update for that provider.

```
POST /api/providers/:id
```

```
{
  "action" : "edit" ,
  "resource" : {
    "hostname" : "new_vcenter_50",
    "ipaddress" : "192.168.150.2"
  }
}
```

Note:

type is a restricted attribute and cannot be edited.

Deleting Providers

Deleting a provider is done via the delete method:

```
DELETE /api/providers/:id
```

or via the **delete** action as follows:

```
POST /api/providers/:id

{
  "action" : "delete"
}
```

or by href:

```
POST /api/providers

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/providers/:id" }
  ]
}
```

Refresh Providers

Performing a refresh of providers is done via the **refresh** action.

```
POST /api/providers/:id

{
  "action" : "refresh"
}
```

You can also refresh multiple providers:

```
POST /api/providers

{
  "action" : "refresh",
  "resources" : [
    { "href" : "http://localhost:3000/api/providers/:id" },
    { "href" : "http://localhost:3000/api/providers/:id" }
  ]
}
```

Import Vm to Provider

Importing a VM into a provider is supported via the *import_vm* action.

This is currently supported for importing a Vm from a VMware vCenter provider into a Red Hat Virtualization provider.

```
POST /api/providers/:id

{
  "action" : "import_vm"
  "resource" : {
    "source" : { "href" : "http://localhost:3000/api/vms/11" },
    "target" : {
      "name"       : "new_vm_name",
      "cluster"    : { "href" : "http://localhost:3000/api/clusters/201" },
      "data_store" : { "href" : "http://localhost:3000/api/data_stores/301" },
      "sparse"     : true
    }
  }
}
```

Foreman Support

Management of Foreman providers is provided as a seperate class of providers. This is still accessed via the **/api/providers** entry point, but enabled by a new parameter:

```
  GET|POST|DELETE /api/providers?provider_class=provider
```

The **provider_class** parameter is supported with **/api/provider** for all CRUD operations including the **refresh** action.

An example of creating a Foreman provider:

```
POST /api/providers?provider_class=provider

{
  "type"      : "ManageIQ::Providers::Foreman::Provider",
  "name"      : "new_foreman_111",
  "url"       : "100.200.300.111",
  "credentials" : {
    "userid"   : "foreman_admin",
    "password" : "foreman_password"
```

```
  }
}
```

When creating providers with provider_class as *provider*, the **type** attribute specifies the supported provider class names which include:

ManageIQ::Providers::AnsibleTower::Provider
ManageIQ::Providers::Foreman::Provider
ManageIQ::Providers::Openstack::Provider

Note:

When managing providers of *provider* provider_class, Tagging and Policy management does not currently apply. So requests including the tags, policies and policy_profiles subcollections will be rejected.

Provider Examples

Below you will find example requests for creating various providers.

Ansible Tower Automation Manager:

```
{
  "type": "ManageIQ::Providers::AnsibleTower::AutomationManager",
  "name": "API test",
  "api_version": "3.1.2",
  "provider_id": 10000000000012,
  "credentials": {
    "userid": "userid",
    "password": "password"
  }
}
```

AWS:

```
{
  "type": "ManageIQ::Providers::Amazon::CloudManager",
  "provider_region": "us-east-1",
  "name": "Amazon Example",
  "credentials": {
    "userid": "<access key id>",
    "password": "<secret access key>"
  }
}
```

Azure:

```
{
  "type": "ManageIQ::Providers::Azure::CloudManager",
  "name": "Azure Example",
  "provider_region": "eastus",
  "uid_ems": "<tenant_id>",
  "subscription": "<subscription_id>",
  "credentials": {
    "userid": "<client id>",
    "password": "<client key>"
  }
}
```

Kubernetes Container Manager:

```
{
  "type": "ManageIQ::Providers::Kubernetes::ContainerManager",
  "name": "Kubernetes Test",
  "connection_configurations": [
    {
      "endpoint": {
        "hostname": "00.0.00.00",
        "port": "6443",
        "security_protocol": "ssl"
      },
      "authentication": {
        "authtype": "bearer",
        "auth_key": "<token_here>"
      }
    }
  ]
}
```

Microsoft System Center VMM (SCVMM):

```
{
  "type": "ManageIQ::Providers::Microsoft::InfraManager",
  "name": "SCVMM API example",
  "api_version": "2.1.0",
  "connection_configurations": [
    {
      "endpoint": {
        "hostname": "00.0.00.00",
        "port": "443",
        "security_protocol": "ssl"
      },
      "authentication": {
        "username": "username",
        "password": "password"
      }
    }
  ]
}
```

OpenShift Container Manager:

```
{
   "name":"ocp-example",
   "port":8443,
   "type":"ManageIQ::Providers::Openshift::ContainerManager",
   "hostname":"ocp.example.com",
   "connection_configurations":[
      {
         "endpoint":{
            "role":"default",
            "security_protocol":"ssl-without-validation",
            "verify_ssl":0
         },
         "authentication":{
            "authtype":"bearer",
            "type":"AuthToken",
            "auth_key":"<auth key>"
         }
      }
   ]
}
```

OpenStack Cloud Manager:

```
{
  "type": "ManageIQ::Providers::Openstack::CloudManager",
  "name": "API Example",
  "api_version": "v2",
  "connection_configurations": [
    {
      "endpoint": {
        "role": "default",
        "hostname": "00.0.00.001",
        "port": "5000",
        "security_protocol": "non-ssl"
      },
      "authentication": {
        "type": "ManageIQ::Providers::Openstack::CloudManager::AuthKeyPair",
        "fingerprint": "fingerprint"
      }
    }
  ]
}
```

OpenStack Infra Manager:

```
{
  "type": "ManageIQ::Providers::Openstack::InfraManager",
  "name": "OSP API example",
  "api_version": "v2",
  "connection_configurations": [
    {
      "endpoint": {
        "hostname": "00.0.00.00",
        "port": "5000",
        "security_protocol": "non-ssl"
      },
      "authentications": {
        "username": "username",
        "password": "password"
      }
```

```
    }
  ]
}
```

Red Hat Virtualization Manager:

```
{
  "type": "ManageIQ::Providers::Redhat::InfraManager",
  "name": "Sample Rhevm",
  "port": 5000,
  "hostname": "sample_rhevm.provider.com",
  "ipaddress": "100.200.300.4",
  "security_protocol": "kerberos",
  "certificate_authority": "----BEGIN CERTIFICATE----...",
  "credentials": {
    "userid": "uname",
    "password": "password"
  }
}
```

VMware Infra Manager:

```
{
  "type": "ManageIQ::Providers::Vmware::InfraManager",
  "name": "Sample VMWare provider",
  "hostname": "sample_vmware.provider.com",
  "ipaddress": "100.200.300.1",
  "credentials": {
    "userid": "uname",
    "password": "pword"
  }
}
```

VMware Cloud Manager:

```
{
  "type": "ManageIQ::Providers::Vmware::CloudManager",
  "api_version": "9.0",
  "name": "Api Test",
  "connection_configurations": [
    {
      "endpoint": {
        "port": "443",
        "hostname": "00.0.00.000"
      },
      "authentication": {
        "username": "username",
        "password": "pword"
      }
    }
  ]
}
```


- Provision Requests
  - Creating Requests
  - Provision Requests Attribute Groups
  - Editing Requests
    - Response:
  - Approving or Denying Requests
    - Approving a Request
    - Denying a Request
  - Monitoring Requests
    - Response:
  - Editing Request Tasks

Provision Requests

Provisioning requests are made available via the following entrypoint.

`/api/provision_requests`

- Creating Requests
- Editing Requests
- Approving or Denying Requests

- Monitoring Requests
- Editing Request Tasks

Creating Requests

Either by specifying a *create* action or by posting the request directly to **/api/provision_requests**.

As an example:

```
{
  "version" : "1.1",
  "template_fields" : {
    "guid" : "afe6e8a0-89fd-11e3-b6ac-b8e85646e742"
  },
  "vm_fields" : {
    "number_of_cpus" : 1,
    "vm_name" : "aab_test_vm1",
    "vm_memory" : "1024",
    "vlan" : "nic1"
  },
  "requester" : {
    "user_name" : "jdoe",
    "owner_first_name" : "John",
    "owner_last_name" : "Doe",
    "owner_email" : "jdoe@sample.com",
    "auto_approve" : true
  },
  "tags" : {
    "network_location" : "Internal",
    "cc" : "001"
  },
  "additional_values" : {
    "request_id" : "1001"
  },
  "ems_custom_attributes" : { },
  "miq_custom_attributes" : { }
}
```

Provision Requests Attribute Groups

Attribute Group	Type	Description
version	string	Interface version. Defaults to 1.1
template_fields	hash	Fields used to find template virtual machine. Provide any or all fields. Supply a guid or ems_guid to protect against matching same-named templates on different Providers within the appliance. Supported fields are: name=[VM Template Name] Example: template_1 guid=[guid value from vms resource] ems_guid=[uid_ems value from vms resource]
vm_fields	hash	Allows for the setting of properties on the Catalog, Hardware, Network, Customize, and Schedule tabs in the Provisioning dialog
requester	hash	Allows for the setting of properties on the Requester tab in the Provisioning dialog
tags	hash	Tags to apply to newly created VM Example: network_location=Internal cc=001
additional_values	hash	Additional values are name-value pairs stored with a provision request, but not used by the core provisioning code. These values are usually referenced from automate methods for custom processing Example: Store a request_id from an external system so the system can be notified during the provisioning process
ems_custom_attributes	hash	Custom attributes applied to the virtual machine through the Provider as part of provisioning
miq_custom_attributes	hash	Custom attributes applied to the virtual machine and stored in the database as part of provisioning

Editing Requests

Updating provision requests can be performed by posting an *edit* action as follows:

```
POST /api/provision_requests/11
```

```
{
  "action" : "edit",
  "resource" : {
    "options" : {
      "custom_var1" : "custom value1"
    }
```

```
  }
}
```

Response:

```
{
  "href": "http://localhost:3000/api/provision_requests/11",
  "id": "11",
  "description": "Provision from [template1] to [###]",
  "approval_state": "pending_approval",
  "type": "MiqProvisionRequest",
  "created_on": "2015-04-14T17:36:30Z",
  "updated_on": "2015-04-14T18:32:30Z",
  "requester_id": "88913",
  "requester_name": "API User",
  "request_type": "template",
  "request_state": "pending",
  "message": "VM Provisioning - Request Created",
  "status": "Ok"
  "options": {
    "use_pre_dialog": false,
    "request_type": "template",
    "miq_request_dialog_name": "miq_provision_dialogs",
    "src_vm_id": [
      109996,
      "template1"
    ],
    "src_vm_nics": [],
    "src_vm_lans": [],
    "src_ems_id": [
      59136,
      "ems_0000000000002"
    ],
    "placement_auto": [
      true,
      1
    ],
    "vm_tags": [],
    "ws_values": {
    },
    "ws_ems_custom_attributes": {
    },
    "ws_miq_custom_attributes": {
    },
    "tags": {
    }
    "userid": "admin",
    "source_id": "109996",
    "source_type": "VmOrTemplate",
    "custom_var1" : "custom value1"
  }
}
```

Approving or Denying Requests

If a provision is not auto approved, it can be approved or denied via the actions *approve* and *deny* on the provision request resource.

Approving a Request

```
POST /api/provision_requests/:id

{
  "action" : "approve",
  "reason" : "reason for approval"
}
```

Denying a Request

```
POST /api/provision_requests/:id

{
  "action" : "deny",
  "reason" : "reason for denial"
}
```

Monitoring Requests

Once a provision request is created, the response result will include the queryable provision request itself, i.e. **/api/provision_requests/:id**

Response:

```
{
  "results": [
    {
      "id": "3068",
      "description": "Provision from [template1] to [###]",
      "approval_state": "pending_approval",
      "type": "MiqProvisionRequest",
      "created_on": "2015-04-14T17:36:30Z",
      "updated_on": "2015-04-14T17:36:30Z",
      "requester_id": "88913",
      "requester_name": "API User",
      "request_type": "template",
      "request_state": "pending",
      "message": "VM Provisioning - Request Created",
      "status": "Ok"
      "options": {
        "use_pre_dialog": false,
        "request_type": "template",
        "miq_request_dialog_name": "miq_provision_dialogs",
        "src_vm_id": [
          109996,
          "template1"
        ],
        "src_vm_nics": [],
        "src_vm_lans": [],
        "src_ems_id": [
          59136,
          "ems_0000000000002"
        ],
        "placement_auto": [
          true,
          1
        ],
        "vm_tags": [],
        "ws_values": {
        },
        "ws_ems_custom_attributes": {
        },
        "ws_miq_custom_attributes": {
        },
        "tags": {
        }
      },
      "userid": "admin",
      "source_id": "109996",
      "source_type": "VmOrTemplate"
    }
  ]
}
```

In the above example, the request could be queried periodically until the **request_state** reaches the **finished** state. Specifically,

```
GET /api/provision_requests/3068
```

Note:

The requests tasks of a provisioning request can also be queried by expanding the request_tasks subcollection as follows:

```
GET /api/provision_requests/:id?expand=request_tasks
```

or by fetching them directly as follows:

```
GET /api/provision_requests/:id/request_tasks?expand=resources
```

Editing Request Tasks

Request tasks for Provision Requests can be updated as follows:

```
POST /api/provision_requests/:id/request_tasks/:request_task_id
```

```
{
  "action" : "edit",
  "resource" : {
    "options" : {
      "request_param_a" : "value_a",
      "request_param_b" : "value_b"
    }
  }
}
```

For a list of attributes available for the provision request attribute groups please refer to the Provision Request Supported Attributes in the appendices section.

- Request Support
  - Creating Requests
  - Querying Requests
  - Editing Requests
  - Approving or Denying Requests
    - Approving a Request
    - Denying a Request
  - Request Types

Request Support

Support for Requests in the API is provided and implemented by specific request types as listed here:

API Collection	Request Type
/api/automation_requests	Automation Requests
/api/provision_requests	Provision Requests
/api/service_requests	Service Requests

A lower level support for creating and updating requests is also provided by the following:

`/api/requests`

- Creating Requests
- Querying Requests
- Editing Requests
- Approving or Denying Requests
- Request Types

Creating Requests

Either by specifying a *create* action or by posting the request directly to **/api/requests**.

As an example:

```
POST /api/requests
```

```
{
  "options" : {
    "request_type" : "service_reconfigure",
    "src_id" : "4",
    "service_param1" : "updated_value1",
    "service_param2" : "updated_value2"
  },
  "requester" : {
    "user_name" : "admin"
  },
  "auto_approve" : true | false
}
```

Response:

```
{
  "results": [
    {
      "id": "97",
      "description": "Service Reconfigure for: svc4",
      "approval_state": "pending_approval",
      "type": "ServiceReconfigureRequest",
      "created_on": "2016-11-10T21:39:08Z",
      "updated_on": "2016-11-10T21:39:08Z",
      "requester_id": "1",
      "requester_name": "Administrator",
      "request_type": "service_reconfigure",
      "request_state": "pending",
      "message": "Service Reconfigure - Request Created",
      "status": "Ok",
      "options": {
        "request_type": "service_reconfigure",
        "src_id": "4",
```

```
        "service_param1": "updated_value1",
        "service_param2": "updated_value2"
      },
      "userid": "admin",
      "source_id": "4",
      "source_type": "Service",
      "tenant_id": "1",
      "process": true
    }
  ]
}
```

Querying Requests

Once a request is created, the response result as shown above will include the id of request itself.

In the above example, the request could be queried periodically until the **request_state** reaches the **finished** state.

```
GET /api/requests/97
```

NOTE: The requests tasks of a request can also be queried by expanding the request_tasks subcollection:

```
GET /api/requests/:id?expand=request_tasks
```

Editing Requests

Updating requests is provided by specifying an *edit* action as follows:

```
POST /api/requests/97
```

```
{
  "action" : "edit",
  "resource" : {
    "options" : {
      "service_param1" : "new_updated_value1",
      "service_param2" : "new_updated_value2"
    },
    "requester" : {
      "user_name" : "admin"
    }
  }
}
```

Response:

```
{
  "href": "http://localhost:3000/api/requests/97",
  "id": "97",
  "options": {
    "request_type": "service_reconfigure",
    "src_id": "4",
    "service_param1": "new_updated_value1",
    "service_param2": "new_updated_value2"
  },
  "description": "Service Reconfigure for: svc3",
  "approval_state": "pending_approval",
  "type": "ServiceReconfigureRequest",
  "created_on": "2016-11-10T21:39:08Z",
  "updated_on": "2016-11-10T22:11:03Z",
  "requester_id": "1",
  "requester_name": "Administrator",
  "request_type": "service_reconfigure",
  "request_state": "pending",
  "message": "Service Reconfigure - Request Created",
  "status": "Ok",
  "userid": "admin",
  "source_id": "4",
  "source_type": "Service",
  "tenant_id": "1",
  "process": true
}
```

Approving or Denying Requests

If a request is not auto approved, it can be approved or denied via the actions *approve* and *deny* on the request resource.

Approving a Request

```
POST /api/requests/:id
```

```
{
  "action" : "approve",
  "reason" : "reason for approval"
}
```

Denying a Request

```
POST /api/requests/:id
```

```
{
  "action" : "deny",
  "reason" : "reason for denial"
}
```

Request Types

The details for creating each type of request using the lower level API, including validation of *request_type* and necessary options is driven by the model.

The basic structure for creating the request is as follows:

```
{
  "options" : {
    "request_type" : "...",
    ...
  },
  "requester" : {
    "user_name" : "..."
  },
  "auto_approve" : true | false
}
```

The list of supported *request_type* values as of this writing is shown here below:

request_type	Description
automation	Automation
clone_to_service	Service Provision
clone_to_template	VM Publish
clone_to_vm	VM Clone
host_pxe_install	Host Provision
provision_via_foreman	Foreman Provision
service_reconfigure	Service Reconfigure
template	VM Provision
vm_migrate	VM Migrate
vm_reconfigure	VM Reconfigure

- Reports Support
  - Querying Reports
  - Running Reports
    - Request:
    - Response:
  - Scheduling Reports
    - Example: “One off” report
      - Request
      - Response
  - Requesting Report Result Downloads
    - Request:
    - Response:
  - Importing Reports
    - Request:
    - Response:

Reports Support

Support for querying, importing and running Reports is now available. Access to reports and report results is through the following collections:

```
/api/reports
```

```
/api/results
```


- Querying Reports

- Running Reports
- Scheduling Reports
- Requesting Report Result Downloads
- Importing Reports

Querying Reports

Querying Available Reports

```
GET /api/reports
```

Querying a Single Report

```
GET /api/reports/:id
```

Querying Results of a Particular Reports

```
GET /api/reports/:id/results
```

Optionally, expanding all results

```
GET /api/reports/:id/results?expand=resources
```

Querying a single result of a report

```
GET /api/reports/:id/results/:result_id
```

Querying report results

```
GET /api/results
```

Running Reports

Running reports is done via the **run** action on a report resource as follows:

```
POST /api/reports/:id
```

```
{
  "action" : "run"
}
```

Example for running a report on the following:

```
{
  "href": "http://localhost:3000/api/reports/80",
  "id": "80",
  "name": "VM Relationships"
}
```

Request:

```
POST /api/reports/80
```

```
{
  "action" : "run"
}
```

Response:

```
{
  "success": true,
  "message": "running report 80",
  "href": "http://localhost:3000/api/reports/80",
  "result_id": "27",
  "result_href": "http://localhost:3000/api/results/27",
  "task_id": "726",
  "task_href": "http://localhost:3000/api/tasks/726"
}
```

The report run's task can be queried periodically via the task_href specified above as follows:

```
GET /api/tasks/726
```

```
{
  "href": "http://localhost:3000/api/tasks/726",
  "id": "726",
  "name": "Generate Report: 'VM Relationships'",
```

```
  "state": "Queued",
  "status": "Ok",
  "message": "Task has been queued",
  "userid": "system",
  "created_on": "2015-11-09T18:44:41Z",
  "updated_on": "2015-11-09T18:44:41Z"
}

{
  "href": "http://localhost:3000/api/tasks/726",
  "id": "726",
  "name": "Generate Report: 'VM Relationships'",
  "state": "Finished",
  "status": "Ok",
  "message": "Generating report complete",
  "userid": "system",
  "created_on": "2015-11-09T18:44:41Z",
  "updated_on": "2015-11-09T18:47:13Z"
}
```

With the results queried via the result_href provided above:

```
GET /api/results/27

{
  "href": "http://localhost:3000/api/results/27",
  "id": "27",
  "name": "VM Relationships",
  "miq_report_id": "80",
  "miq_task_id": "726",
  "userid": "system",
  "report_source": "Requested by user",
  "db": "Vm",
  "report": {
    "name": "VM Relationships",
    "title": "VM Relationships",
    "rpt_group": "Relationships - Virtual Machines, Folders, Clusters",
    "rpt_type": "Default",
    "priority": 10,
    "db": "Vm",
    "cols": [
      "name",
      "v_owning_datacenter",
      "v_owning_folder",
      "v_owning_resource_pool"
    ],
    "include": {
      "ext_management_system": {
        "columns": [
          "name"
        ]
      },
      "host": {
        "columns": [
          "v_owning_cluster",
          "hostname"
        ]
      },
      "storage": {
        "columns": [
          "name"
        ]
      }
    },
    "col_order": [
      "name",
      "ext_management_system.name",
      "v_owning_datacenter",
      "host.v_owning_cluster",
      "host.hostname",
      "v_owning_folder",
      "v_owning_resource_pool",
      "storage.name"
    ],
    "headers": [
      "VM Name",
      "Provider Name",
      "Datacenter",
      "Cluster",
      "Host",
      "Folder",
      "Resource Pool",
```

```
      "Datastore"
    ],
    "conditions": {
      "exp": {
        "IS NOT EMPTY": {
          "field": "Vm.host-name"
        }
      },
      "col_details": {
        "Vm.host-name": {
          "data_type": "string",
          "virtual_reflection": false,
          "virtual_column": false,
          "excluded_by_preprocess_options": false,
          "tag": false,
          "include": {
            "host": {
            }
          },
          "format_sub_type": "string"
        }
      },
      "pexp": {
        "IS NOT EMPTY": {
          "field": "Vm.host-name"
        }
      },
      "ruby": "<value ref=vm, type=string>/virtual/host/name</value> != \"\""
    },
    "order": "Ascending",
    "sortby": [
      "name"
    ],
    "filename": "450_Relationships - Virtual Machines, Folders, Clusters/010_VMs Relationships.yaml",
    "file_mtime": "2015-08-19T13:23:50Z",
    "template_type": "report",
    "col_formats": [

    ],
    "rpt_options": {
    },
    "extras": {
      "total_html_rows": 1709
    },
    "user_categories": [

    ],
    "trend_data": {
    }
  },
  "created_on": "2015-11-09T18:52:48Z",
  "last_run_on": "2015-11-09T18:52:56Z",
  "last_accessed_on": "2015-11-09T18:52:56Z",
  "result_set": [
    {
      "name": "aab-vm1",
      "v_owning_datacenter": "",
      "v_owning_folder": "",
      "v_owning_resource_pool": "",
      "host.v_owning_cluster": "",
      "host.hostname": "test-host.testdomain.com",
      "storage.name": "StarStorage1"
      "id": "320"
    },
    {
      "name": "aab_demo_vm",
      "v_owning_datacenter": "",
      "v_owning_folder": "",
      "v_owning_resource_pool": "",
      "host.v_owning_cluster": "",
      "host.hostname": "test-host2.testdomain.com",
      "storage.name": "StarStorage2",
      "id": "1"
    },
    ...
  ]
}
```

Note that one can query all the results run for a report as follows:

```
GET /api/reports/80/results?expand=resources&attributes=created_on
```

```
{
  "name": "results",
  "count": 27,
  "subcount": 2,
  "resources": [
    {
      "href": "http://localhost:3000/api/reports/80/results/27",
      "id": "27",
      "created_on": "2015-11-09T18:52:48Z"
    },
    {
      "href": "http://localhost:3000/api/reports/80/results/13",
      "id": "13",
      "created_on": "2015-07-29T18:24:35Z"
    }
  ]
}
```

Scheduling Reports

Scheduling a report is done via the **schedule** action on a report resource as follows

```
POST /api/reports/:id
```

```
{
  "action" : "schedule"
  "resource": {
    "send_email": true|false,
    "start_date": DateTime,
    "time_zone":  TZ,
    "interval": {
      "unit":  "minutely"|"hourly"|"daily"|"weekly"|"monthly"|"once",
      "value": "once"
    },
    ...
  }
}
```

Example: “One off” report

Request

```
POST /api/reports/5
```

```
{
  "action": "schedule",
  "resource": {
    "name":        "Chargeback Report Schedule - 2000/01/01",
    "description": "Chargeback Report 2000 New Year Report",
    "resource_type": "MiqReport",
    "start_date": "2000-01-01T00:00:00Z",
    "time_zone": "UTC",
    "interval": { "unit": "once" },
    "enabled": true
  }
}
```

Response

```
{
  "success":       true,
  "message":       "scheduling of report 5",
  "schedule_id":   "10",
  "schedule_href": "http://localhost:3000/api/reports/5/schedules/10",
  "href":          "http://localhost:3000/api/reports/5"
}
```

Requesting Report Result Downloads

Requesting a **txt**, **csv** or **pdf** download of the report results can be done via the **request_download** action requiring the desired *result_type* as follows:

Request:

```
POST /api/results/22
```

```
{
  "action" : "request_download",
```

```
  "result_type" : "csv"
}
```

This triggers an async request of the report and returns an action response as follows:

Response:

```
{
  "success": true,
  "message": "Requesting a download of a csv report for Result id:22 name:VMware VM Summary",
  "task_id": "46",
  "task_href": "http://localhost:3000/api/tasks/46",
  "task_results_href": "http://localhost:3000/api/tasks/46/task_results"
}
```

The task can simply be queried for a *Finished* state and the results then fetched via the *task_results_href* provided above.

Querying for the task completion:

```
GET /api/tasks/46
```

```
{
  "href": "http://localhost:3000/api/tasks/46",
  "id": "46",
  "name": "Generate Report result [csv]: 'VMware VM Summary'",
  "state": "Finished",
  "status": "Ok",
  "message": "Generate Report result [csv]",
  "userid": "admin",
  "created_on": "2020-05-06T17:15:29Z",
  "updated_on": "2020-05-06T17:15:30Z",
  "pct_complete": null,
  "context_data": {
    "result_id": "22",
    "result_type": "csv",
    "session_id": "1cae5e9c-9b3e-4ee1-a286-bc2a0131233d-7f4bc905"
  },
  "results": null,
  "miq_server_id": "3",
  "identifier": null,
  "started_on": "2020-05-06T17:15:30Z",
  "zone": null,
  "actions": [
    {
      "name": "delete",
      "method": "post",
      "href": "http://localhost:3000/api/tasks/46"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "http://localhost:3000/api/tasks/46"
    }
  ]
}
```

And the results can be fetched as follows:

```
GET /api/tasks/46/task_results
```

This will return the report with the appropriately typed **Content-Type**, *application/txt*, *application/csv*, or *application/pdf* as per the requested download.

With a **request_download** action targeted at the collection level, one can also request the download for multiple report results as follows:

```
POST /api/results
```

```
{
  "action" : "request_download",
  "resources" : [
      { "id" : "15", "result_type" : "txt" },
      { "id" : "17", "result_type" : "txt" },
      { "id" : "22", "result_type" : "csv" }
   ]
}
```

```
{
  "results" : [
    {
      "success": true,
      "message": "Requesting a download of a txt report for Result id:15 name:VMware VM Summary",
```

```
      "task_id": "54",
      "task_href": "http://localhost:3000/api/tasks/54",
      "task_results_href": "http://localhost:3000/api/tasks/54/task_results"
    },
    {
      "success": true,
      "message": "Requesting a download of a txt report for Result id:17 name:VMware VM Summary",
      "task_id": "55",
      "task_href": "http://localhost:3000/api/tasks/55",
      "task_results_href": "http://localhost:3000/api/tasks/55/task_results"
    },
    {
      "success": true,
      "message": "Requesting a download of a txt report for Result id:22 name:VMware VM Summary",
      "task_id": "56",
      "task_href": "http://localhost:3000/api/tasks/56",
      "task_results_href": "http://localhost:3000/api/tasks/56/task_results"
    }
  ]
}
```

Each can then be individually queried and downloaded upon completion.

Importing Reports

Importing reports to the system can be done by posting an **import** action to the reports collection as follows:

```
POST /api/reports

{
  "action" : "import",
  "resource" : {
    "report" : { JSON serialized report },
    "options" : { "save" : true }
  }
}
```

Here's an example:

Request:

```
POST /api/reports

{
  "action" : "import",
  "resource" : {
    "report" : {
      "menu_name" : "Test Report",
      "col_order" : ["col1", "col2", "col3"],
      "cols"      : ["col1", "col2", "col3"],
      "rpt_type"  : "Custom",
      "title"     : "Test Report",
      "db"        : "My::Db",
      "rpt_group" : "Custom"
    },
    "options" : { "save" : true }
  }
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Imported Report: [Test Report]",
      "result": {
        "id": "199",
        "name": "Test Report",
        "title": "Test Report",
        "rpt_group": "Custom",
        "rpt_type": "Custom",
        "db": "My::Db",
        "cols": [
          "col1",
          "col2",
          "co3l"
        ],
        "col_order": [
          "col1",
```

```
          "col2",
          "col3"
        ],
        "created_on": "2015-11-09T19:44:37Z",
        "updated_on": "2015-11-09T19:44:37Z",
        "miq_group_id": "1",
        "user_id": "1"
      }
    }
  ]
}
```


- Role Management
  - Querying Roles
  - Creating Roles
  - Editing Roles
    - Assigning features to a role
    - Unassigning features from a role
  - Deleting Roles

Role Management

Management of User Roles is provided via the following collection:

`/api/roles`

as well as the associated product features via:

`/api/features`

Full CRUD actions on roles is available:

- Querying Roles
- Creating Roles
- Editing Roles
- Deleting Roles

Querying Roles

Querying all roles in the system is simply:

`GET /api/roles`

Getting details on a specific role:

`GET /api/roles/:id`

Including the features it entitles:

`GET /api/roles/:id?expand=features`

Or simply querying the features entitled by the role:

`GET /api/roles/:id/features?expand=resources`

Querying all product features of the system that can be assigned to a role.

`GET /api/features`

Creating Roles

Roles can be created via a POST to the roles collection or via the **create** action signature which also allows creation of multiple roles in a single request.

`POST /api/roles`

```
{
  "action" : "create",
  "resource" : {
    "name" : "sample_role",
    "settings" : { "restrictions" : { "vms" : "user" } },
    "features" : [
      { "identifier" : "vm_explorer" },
      { "identifier" : "ems_infra_tag" },
```

```
      { "identifier" : "miq_report_run" }
    ]
  }
}
```

Note:

restrictions for **vms** can be either *user* or *user_or_group*

features can be specified via **identifier**, **href**, or **id**

or creating multiple roles:

```
{
  "action" : "create",
  "resources" : [
    { "name" : "sample_role1", ... },
    { "name" : "sample_role2", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Roles.

Editing Roles

```
POST /api/roles/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "name" : "updated_sample_role",
    "settings" : { "restrictions" : { "vms" : "user_or_group" } }
  }
}
```

or editing multiple roles:

```
POST /api/roles
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/roles/101",
      "name" : "updated_sample_role1"
    },
    {
      "href" : "http://localhost:3000/api/roles/102",
      "name" : "updated_sample_role2"
    },
    ...
  ]
}
```

Assigning features to a role

```
POST /api/roles/:id/features
```

```
{
  "action" : "assign",
  "resource" : {
    "identifier" : "miq_request_view"
  }
}
```

or assigning multiple features:

```
{
  "action" : "assign",
  "resources" : [
    { "identifier" : "miq_request_view" },
    { "identifier" : "storage_manager_show_list" },
    ...
  ]
}
```

Unassigning features from a role

```
POST /api/roles/:id/features
```

```
{
  "action" : "unassign",
  "resource" : {
    { "identifier" : "miq_request_view" }
  }
}
```

or unassigning multiple features:

```
{
  "action" : "unassign",
  "resources" : [
    { "identifier" : "miq_request_view" },
    { "identifier" : "storage_manager_show_list" },
    ...
  ]
}
```

Deleting Roles

Non system roles (i.e. read_only false) can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/roles/101
```

```
{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/roles/101
```

Deleting multiple roles can be done as follows:

```
POST /api/roles
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/roles/101" },
    { "href" : "http://localhost:3000/api/roles/102" },
    ...
  ]
}
```


- Security Groups
  - Querying Security Groups
  - Managing Security Groups
  - Add Security Group
  - Remove Security Group

Security Groups

Management of Security Groups is available via the following collection:

```
/api/security_groups
```

Querying Security Groups

Querying all available Security Groups:

```
GET /api/security_groups
```

Getting details on a specific Security Group:

```
GET /api/security_groups/:id
```

Querying the Security Groups of a specific resource is also possible for the following collections:

Collection
/api/cloud_tenants
/api/instances
/api/providers
/api/vms

```
GET /api/cloud_tenants/:id/security_groups
GET /api/cloud_tenants/:id/security_groups/:security_group_id

GET /api/instances/:id/security_groups
GET /api/instances/:id/security_groups/:security_group_id

GET /api/providers/:id/security_groups
GET /api/providers/:id/security_groups/:security_group_id

GET /api/vms/:id/security_groups
GET /api/vms/:id/security_groups/:security_group_id
```

Managing Security Groups

Management of Security Groups is available for Instances with the following actions:

Action	Description
add	Add security group(s) to the Instance
remove	Remove security group(s) from the Instance

Add Security Group

Security Groups can be added to Instances individually:

```
POST /api/instances/:id/security_groups
```

```
{
  "action" : "add",
  "resource" : {
    "name" : "security_group_a"
  }
}
```

Or in bulk as follows:

```
{
  "action" : "add",
  "resources" : [
    { "name" : "security_group_a" },
    { "name" : "security_group_b" }
  ]
}
```

Remove Security Group

Security groups can be removed from Instances individually:

```
POST /api/instances/:id/security_groups
```

```
{
  "action" : "remove",
  "resource" : {
    "name" : "security_group_c"
  }
}
```

Or in bulk as follows:

```
{
  "action" : "remove",
  "resources" : [
    { "name" : "security_group_d" },
    { "name" : "security_group_e" },
    { "name" : "security_group_f" }
  ]
}
```


- Service Management
  - Service Catalogs
    - Creating Service Catalogs
    - Editing Service Catalogs
    - Assigning Service Templates
    - Ordering Services Service Templates
    - Refresh Service Dialog Fields
      - Refresh Dialog Fields of Service Templates Service Dialogs
      - Refresh Dialog Fields of Service Dialogs
    - Deleting Service Catalogs

- Service Templates
  - Creating Service Templates
  - Editing Service Templates
  - Deleting Service Templates
- Services ===
  - Creating Services ====
  - Service Actions ====
  - Service Editing
  - Adding and Removing Resources from Services
  - Adding Provider VMs to Services
  - Requesting a Chargeback Report for a Service
  - Service Retiring (now)
  - Service Retiring (future)
  - Service Reconfiguring
  - Starting Services
  - Stopping Services
  - Suspending Services
  - Deleting Services
- Service Requests ===
  - Querying Service Requests
  - Approving or Denying Service Requests
    - Approving a Service Request
  - Denying a Service Request
  - Adding Approvers to Service Requests
  - Removing Approvers from Service Requests
- Editing Request Tasks

Service Management

Service management is provided via the following collections:

Path	Description
/api/service_catalogs	Service Catalogs
/api/service_templates	Service Templates
/api/services	Services
/api/service_requests	Service Requests

Service Catalogs

Full CRUD is available on service catalogs.

- Creating Service Catalogs
- Editing Service Catalogs
- Assigning/Unassigning Service Templates from Service Catalogs
- Ordering Services
- Refresh Service Dialog Fields
- Deleting Service Catalogs

Creating Service Catalogs

`/api/service_catalogs`

Creating service catalog is provided via the following actions:

Method	Versions	Comment
create	2.0.0	Preferred
add	since v1.0	

Example of creating service catalog with *create* method:

```
POST /api/service_catalogs
```

```
{
   "action" : "create",
   "resource" : {
     "name" : "Sample Service Catalog",
     "description" : "Description of Sample Service Catalog",
     "service_templates" : [
       { "href" : "http://localhost:3000/api/service_templates/3" },
       { "href" : "http://localhost:3000/api/service_templates/4" }
     ]
   }
}
```

Optionally without the create action signature:

```
{
  "name" : "Sample Service Catalog",
  "description" : "Description of Sample Service Catalog",
  "service_templates" : [
    { "href" : "http://localhost:3000/api/service_templates/3" },
    { "href" : "http://localhost:3000/api/service_templates/4" }
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Service Catalogs.

Editing Service Catalogs

Editing a service catalog via the *edit* action as follows:

```
POST /api/service_catalogs/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "Updated Description of Third Service Catalog"
   }
}
```

Assigning Service Templates

Assign and Unassigning service templates to service catalogs is done via the *assign* and *unassign* actions as follows:

```
POST /api/service_catalogs/:id
```

```
{
  "action" : "assign",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_templates/5" },
    { "href" : "http://localhost:3000/api/service_templates/6" }
  ]
}
```

```
POST /api/service_catalogs/:id
```

```
{
  "action" : "unassign",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_templates/1" },
    { "href" : "http://localhost:3000/api/service_templates/2" }
  ]
}
```

Ordering Services Service Templates

Ordering services is done by triggering an *order* action on service templates in a service catalog as follows:

```
POST /api/service_catalogs/:id/service_templates
```

```
{
  "action" : "order",
  "resource" : {
    "href" : "http://localhost:3000/api/service_templates/3",
    "option_0_vm_target_name" : "test-vm-0001",
    "option_0_vm_target_hostname" : "test-vm-0001"
  }
}
```

Refresh Service Dialog Fields

Refreshing dialog fields is done via a *refresh_dialog_fields* action as shown in the following cases:

Refresh Dialog Fields of Service Templates Service Dialogs

Refreshing dialog fields for a service template service dialogs is done as follows:

```
POST /api/service_catalogs/:id/service_templates
```

```
{
  "action" : "refresh_dialog_fields",
  "resource" : {
    "href" : "http://localhost:3000/api/service_templates/3",
    "dialog_fields" : {
      "my_field1" : "my field1 value",
      "my_field2" : "my field2 value",
      "my_field3" : "my field3 value",
      ...
    },
    "fields" : [ "my_field1", "my_field3" ]
  }
}
```

Refresh Dialog Fields of Service Dialogs

Refreshing dialog fields for a service dialog is done as follows:

```
POST /api/service_dialogs/:id
```

```
{
  "action" : "refresh_dialog_fields",
  "resource" : {
    "dialog_fields" : {
      "my_field1" : "my field1 value",
      "my_field2" : "my field2 value",
      "my_field3" : "my field3 value",
      ...
    },
    "fields" : [ "my_field1", "my_field3" ]
  }
}
```

Deleting Service Catalogs

Single service catalogs can simply be deleted via the DELETE method as follows:

```
DELETE /api/service_catalogs/:id
```

Multiple service catalogs can also be deleted with a single request as follows:

```
POST /api/service_catalogs
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_catalogs/11" },
    { "href" : "http://localhost:3000/api/service_catalogs/12" },
    { "href" : "http://localhost:3000/api/service_catalogs/13" }
  ]
}
```

Service Templates

Management of Service Templates is available via the following collection:

```
/api/service_templates
```

Full CRUD is available on service templates.

- Creating Service Templates
- Editing Service Templates
- Deleting Service Templates

Creating Service Templates

Service Templates can be created via a POST to the /api/service_templates collection:

```
/api/service_templates
```

```
{
  "name" : "Atomic Service Template",
  "service_type" : "atomic",
  "prov_type"    : "amazon",
  "display"      : "false",
  "config_info"  : {
    "miq_request_dialog_name" : "Dialog_Name",
    "placement_auto"          : [true, 1],
    "number_of_vms"           : [1, "1"],
    "src_vm_id"               : [11, "test_vm"],
    "vm_name"                 : "AtomicVMName",
    "schedule_type"           : ["immediately", "Immediately on Approval"],
    "instance_type"           : [21, "instance_flavor"],
    "src_ems_id"              : [5, "test_provider"],
    "provision"               : {
      "fqname"    : "/Sample/System/ProvisionEndpoint",
      "dialog_id" : "21"
    },
    "retirement"              : {
      "fqname"    : "/Sample/System/RetirementEndpoint",
      "dialog_id" : "22"
    }
  }
}
```

Editing Service Templates

Editing a single service template as follows:

```
POST /api/service_templates/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "name" : "svc_template_01",
    "description" : "This is a sample service template"
  }
}
```

Deleting Service Templates

Deleting single service template via the DELETE method:

```
DELETE /api/service_templates/:id
```

or via the *delete* action as follows:

```
POST /api/service_templates/:id
```

```
{
  "action" : "delete"
}
```

Multiple service templates can also be deleted in bulk as follows:

```
POST /api/service_templates
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_templates/11" },
    { "href" : "http://localhost:3000/api/service_templates/12" }
  ]
}
```

Services ===

Accessing services is via the **services** collection as follows:

```
/api/services
```

or addressing an individual Service via the **services** resource:

```
/api/services/:id
```

Creating Services ====

In general, services are created once ordered from service catalogs.

Service resources can also be created as follows:

```
POST /api/services
```

```
{
  "action" : "create",
  "resource" : {
    "name" : "test_service_name",
    "description" : "Test Service",
    "parent_service" : { "href" : "http://localhost:3000/api/services/15" }
  }
}
```

In the above example *parent_service* is an *href* or *id* reference to an existing service.

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Services.

Service Actions ====

Once services are ordered from a catalog, they can be:

- Edited (Basic)
- Retired now
- Retired in future
- Reconfigured
- Started
- Stopped
- Suspended
- Deleted

Additional Service actions include:

- Adding and Removing Resources from Services
- Adding Provider VMs to Services
- Requesting a Chargeback Report for a Service

Other than service creates and edits, service actions return the action response as in the following example:

Request:

```
POST /api/services/12
```

```
{
  "action" : "start"
}
```

Response:

```
{
  "success": true,
  "message": "Service id:12 name:'SampleService' starting",
  "task_id": "62",
  "task_href": "http://localhost:3000/api/tasks/62",
  "href": "http://localhost:3000/api/services/12"
}
```

Service Editing

```
POST /api/services/:id
```

```
{
   "action" : "edit",
   "resource" : {
     "name" : "Updated Service Name",
     "description" : "Updated Service Description"
   }
}
```

Adding and Removing Resources from Services

Adding and removing resources from services is made available with the following actions:

Action
add_resource
remove_resource
remove_all_resources

The following example adds a single resource to a service:

```
POST /api/services/:id
```

```
{
  "action" : "add_resource",
  "resource" : { "resource" : { "href" : "http://localhost:3000/api/vms/11" } }
}
```

The following example bulk adds resources to multiple services:

```
POST /api/services
```

```
{
   "action" : "add_resource",
   "resources" : [
      {
        "href" : "http://localhost:3000/api/services/3",
        "resource" : { "href" : "http://localhost:3000/api/vms/11" }
      },
      {
        "href" : "http://localhost:3000/api/services/3",
        "resource" : { "href" : "http://localhot:3000/api/vms/12" }
      },
      {
        "href" : "http://localhost:3000/api/services/4",
        "resource" : { "href" : "http://localhost:3000/api/vms/21" }
      },
      {
        "href" : "http://localhost:3000/api/services/5",
        "resource" : { "href" : "http://localhost:3000/api/vms/22" }
      }
   ]
}
```

The following example removes a single resource from a service:

```
POST /api/services/:id
```

```
{
  "action" : "remove_resource",
  "resource" : { "resource" : { "href" : "http://localhost:3000/api/vms/11" } }
}
```

The following example removes resources in bulk from multiple services:

```
POST /api/services
```

```
{
  "action" : "remove_resource",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/services/3",
      "resource" : { "href" : "http://localhost:3000/api/vms/11" }
    },
    {
      "href" : "http://localhost:3000/api/services/3",
      "resource" : { "href" : "http://localhot:3000/api/vms/12" }
    },
    {
      "href" : "http://localhost:3000/api/services/4",
      "resource" : { "href" : "http://localhost:3000/api/vms/21" }
    },
    {
      "href" : "http://localhost:3000/api/services/5",
      "resource" : { "href" : "http://localhost:3000/api/vms/22" }
    }
  ]
}
```

The following example removes all resources from a single service:

```
POST /api/services/:id
```

```
{
  "action" : "remove_all_resources"
}
```

The following example removes all resources in bulk from multiple services:

```
POST /api/services
```

```
{
  "action" : "remove_all_resources",
  "resources" : [
    { "href" : "http://localhost:3000/api/services/11" },
    { "href" : "http://localhost:3000/api/services/12" },
    { "href" : "http://localhost:3000/api/services/13" }
  ]
}
```

Adding Provider VMs to Services

For cases where VMs are created externally, i.e. via Ansible Playbooks, this action allows the system to identify the VMs created and link those VMs back to the service.

```
POST /api/services/:id
```

```
{
  "action" : "add_provider_vms",
  "resource" : {
    "provider" : { "href" : "http://localhot:3000/api/providers/11" }
    "uid_ems" : [
      "provider_vm_id1",
      "provider_vm_id2",
      ...
    ]
  }
}
```

Requesting a Chargeback Report for a Service

This action will request a chargeback report for the service.

```
POST /api/services/:id
```

```
{
  "action" : "queue_chargeback_report"
}
```

The queue_chargeback_report action will return an action response which includes the task reference as shown here:

```
{
  "success": true,
  "message": "Queued chargeback report generation for Service id:1 name:'test_service1'",
  "task_id": "39",
  "task_href": "http://localhost:3000/api/tasks/39"
}
```

The task_href can be queried and upon completion of the task, the chargeback report for the service can be queried as follows:

```
GET /api/services/:id?attributes=chargeback_report
```

Service Retiring (now)

```
POST /api/services/:id
```

```
{
  "action" : "request_retire"
}
```

Service Retiring (future)

```
POST /api/services/:id
```

```
{
  "action" : "request_retire",
  "resource" : { "date" : "10/30/2015", "warn" : "5" }
}
```

Service Reconfiguring

When a service is defined with a Reconfigure Automation Entrypoint, the *reconfigure* action is shown as a valid action for that service.

This action takes any of the dialog fields the service supports during reconfiguration.

```
POST /api/services/:id

{
  "action" : "reconfigure",
  "resource" : {
    "field1" : "updated field1 value",
    "field2" : "updated field2 value",
    ...
  }
}
```

Starting Services

```
POST /api/services/:id

{
  "action" : "start"
}
```

Stopping Services

```
POST /api/services/:id

{
  "action" : "stop"
}
```

Suspending Services

```
POST /api/services/:id

{
  "action" : "suspend"
}
```

Deleting Services

```
DELETE /api/services/:id
```

Or via the delete action for deleting multiple services:

```
POST /api/services

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/services/101",
    { "href" : "http://localhost:3000/api/services/102",
    { "href" : "http://localhost:3000/api/services/103"
    ...
  ]
}
```

Service Requests ===

- Querying Service Requests
- Approving or Denying Service Requests
- Adding Approvers to Service Requests
- Removing Approvers to Service Requests
- Editing Request Tasks

Querying Service Requests

Once services are ordered from a catalog, service requests can be monitored as follows:

```
GET /api/service_requests/:id
```

Note:

The request tasks of service requests can be queried by expanding the request_tasks as follows:

```
GET /api/service_requests/:id?expand=request_tasks
```

or by fetching them directly as follows:

```
GET /api/service_requests/:id/request_tasks?expand=resources
```

Approving or Denying Service Requests

If a service request is not auto approved, it can be approved or denied via the actions *approve* and *deny* on the service request resource.

Approving a Service Request

```
POST /api/service_requests/:id
```

```
{
  "action" : "approve",
  "reason" : "reason for approval"
}
```

Denying a Service Request

```
POST /api/service_requests/:id
```

```
{
  "action" : "deny",
  "reason" : "reason for denial"
}
```

Adding Approvers to Service Requests

Approvers can be added to service requests with the *add_approver* action. Approvers can be specified by *user_id* or *user* reference including an *href* or *id* of the approver.

```
POST /api/service_requests/:id
```

```
{
  "action" : "add_approver",
  "resource" : {
    "user_id" : 11
  }
}
```

Or one of the following signatures:

```
{
  "action" : "add_approver",
  "resource" : {
    "user" : { "id" : 11  }
  }
}
```

```
{
  "action" : "add_approver",
  "resource" : {
    "user" : { "href" : "http://localhost:3000/api/users/11" }
  }
}
```

Approvers can also be added in bulk as follows:

```
POST /api/service_requests
```

```
{
  "action" : "add_approver",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/service_requests/101",
      "user_id" : "11"
    },
    {
      "href" : "http://localhost:3000/api/service_requests/102",
      "user" : { "id" : 11 }
    },
    {
      "href" : "http://localhost:3000/api/service_requests/103",
      "user" : { "href" : "http://localhost:3000/api/users/11" }
    }
  ]
}
```

Removing Approvers from Service Requests

Approvers can be removed from service requests with the *remove_approver* action. Approvers can be specified by *user_id* or *user* reference including an *href* or *id* of the approver.

```
POST /api/service_requests/:id

{
  "action" : "remove_approver",
  "resource" : {
    "user_id" : "11"
  }
}
```

Approvers can also be removed in bulk as follows:

```
POST /api/service_requests

{
  "action" : "remove_approver",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/service_requests/101",
      "user_id" : "11"
    },
    {
      "href" : "http://localhost:3000/api/service_requests/102",
      "user" : { "id" : "11" }
    },
    {
      "href" : "http://localhost:3000/api/service_requests/103",
      "user" : { "href" : "http://localhost:3000/api/users/11" }
    }
  ]
}
```

Editing Request Tasks

Request tasks of Service Requests can be updated as follows:

```
POST /api/service_requests/:id/request_tasks/:request_task_id

{
  "action" : "edit",
  "resource" : {
    "options" : {
      "request_param_a" : "value_a",
      "request_param_b" : "value_b"
    }
  }
}
```


- Service Queries
  - Querying Pictures pertaining to services
  - Querying Provision Dialogs on services and service requests
  - Querying Resource Actions
  - Querying Service and Provision Dialogs
  - Querying Service Dialogs
  - Querying Users that ordered services
  - Querying VMs of Services
  - Querying Orchestration Stacks of Services

Service Queries

Queries to services and related service dialogs are provided via the following collections:

Path	Description
/api/services	Services
/api/service_requests	Service Requests
/api/service_templates	Service Templates
/api/service_dialogs	Service Dialogs

- Querying Pictures
- Querying Provision Dialogs
- Querying Resource Actions

- Querying Service and Provision Dialogs
- Querying Service Dialogs
- Querying Users
- Querying VMs
- Querying Orchestration Stacks

Querying Pictures pertaining to services

Access to pictures for service related collections is made available with the *picture* virtual attribute and related *image_href* attribute.

Querying all resources:

```
GET /api/services?expand=resources&attributes=picture,picture.image_href

GET /api/service_requests?expand=resources&attributes=picture,picture.image_href

GET /api/service_templates?expand=resources&attributes=picture,picture.image_href
```

Querying specific resources:

```
GET /api/services/:id?attributes=picture,picture.image_href

GET /api/service_requests/:id?attributes=picture,picture.image_href

GET /api/service_templates/:id?attributes=picture,picture.image_href
```

Querying Provision Dialogs on services and service requests

Querying provision dialogs on ordered services and related service_requests is provided by the *provision_dialog* virtual attribute:

```
GET /api/services/:id?attributes=provision_dialog

GET /api/service_requests/:id?attributes=provision_dialog
```

Querying Resource Actions

Querying resource_actions is provided as a subcollection of service_templates as follows:

```
GET /api/service_templates/:id/resource_actions
```

This allows queries like:

```
GET /api/service_templates/:id/resource_actions
GET /api/service_templates/:id/resource_actions/:resource_action_id
GET /api/service_templates/:id/resource_actions?expand=resources
GET /api/service_templates/:id/resource_actions?expand=resources&filter[]=action='Provision'
```

Querying Service and Provision Dialogs

Access to service and provision dialogs is provided by the primary collections:

/api/service_dialogs
/api/provision_dialogs

Querying Service Dialogs

Service dialogs can be queried directly from the primary collection:

```
GET /api/service_dialogs
GET /api/service_dialogs/:id
GET /api/service_dialogs/:id?resource_action_id=:ra_id
                            &target_id=:target_id
                            &target_type=:target_type
```

Note:

Above :target_type would be a valid resource type containing the service dialog like *service_template* or *service*.

Or from the Service Template where they are defined and from the related ordered services as follows:

```
GET /api/service_templates/:id/service_dialogs
GET /api/service_templates/:id/service_dialogs/:service_dialog_id
```

```
GET /api/services/:id/service_dialogs
GET /api/services/:id/service_dialogs/:service_dialog_id
```

Querying Users that ordered services

This is made available via the *user* virtual attribute which reflect the user object.

```
GET /api/services/:id?attributes=user
```

```
GET /api/service_requests/:id?attributes=user
```

Optionally, specific user attribute can also be queried as follows:

```
GET /api/services/:id?attributes=user.name,user.email
GET /api/services/:id?attributes=user.name,user.email
```

Querying VMs of Services

This is made available via the *vms* subcollection of services as follows:

```
GET /api/services/:id/vms
```

```
GET /api/services/:id?expand=vms
```

It is also possible to fetch additional attributes for the vms of a service all within the same request by using the *vms.* prefix in the attributes specification as follows:

```
GET /api/services/:id?expand=vms&attributes=vms.num_cpu
```

Accessing decorators for VM resources is also made available here as follows:

```
GET /api/services/:id?expand=vms
                      &attributes=vms.num_cpu
                      &decorators=vms.supports_console?,vms.supports_cockpit?,vms.supports_vnc_console?
```

Querying Orchestration Stacks of Services

This is made available via the *orchestration_stacks* subcollection of services as follows:

```
GET /api/services/:id/orchestration_stacks
```

Fetching the orchestration stacks in addition to the service as follows:

```
GET /api/services/:id?expand=orchestration_stacks
```

Accessing a specific orchestration stack of a service as follows:

```
GET /api/services/:id/orchestration_stacks/:orchestration_stack_id
```


- Service Custom Actions
  - Request:
  - Response:
    - Request:
    - Response:

Service Custom Actions

With button groups, custom buttons and related dialogs defined on Service Templates, the REST API provides visibility to custom buttons on service templates and related ordered services.

This is made available with the *custom_actions* and *custom_action_buttons* virtual attributes on service_templates and services.

- *custom_actions* returns the custom buttons as layed out in the UI on the service. The information returned includes any Resource Actions and related Dialogs for the custom button.
- *custom_action_buttons* returns the array of custom buttons without the custom button groups hierarchy or details on related dialogs.

Fetching custom actions and custom action buttons for a service_template or a deployed service

```
GET /api/service_templates/:id?attributes=custom_actions,custom_action_buttons
```

```
GET /api/services/:id?attributes=custom_actions,custom_action_buttons
```

Example Service Template with custom buttons:

Request:

```
GET /api/service_templates/7?attributes=custom_actions,custom_action_buttons
```

Response:


```
{
  "href": "http://localhost:3000/api/service_templates/7",
  "id": "7",
  "name": "SampleServiceCatalogItem2",
  "description": "Sample Service Catalog Item 2",
  "guid": "47edc73a-4cef-11e5-8525-000a27020067",
  "options": {
    "button_order": [
      "cb-1",
      "cb-2",
      "cbg-43",
      "cb-3"
    ]
  },
  "created_at": "2015-08-27T19:10:29Z",
  "updated_at": "2015-09-16T18:24:13Z",
  "display": true,
  "evm_owner_id": "1",
  "service_type": "atomic",
  "prov_type": "generic",
  "service_template_catalog_id": "7",
  "long_description": "",
  "tenant_id": "1",
  "custom_actions": {
    "buttons": [
      {
        "id": "1",
        "guid": "3d5975fe-57bb-11e5-8d22-000a27020067",
        "description": "Rescan Service Offering",
        "applies_to_class": "ServiceTemplate",
        "options": {
          "button_image": 2,
          "display": true
        },
        "userid": "admin",
        "created_on": "2015-09-10T12:55:40Z",
        "updated_on": "2015-09-16T14:28:17Z",
        "name": "Rescan",
        "visibility": {
          "roles": [
            "_ALL_"
          ]
        },
        "applies_to_id": "7",
        "resource_action": {
          "id": "7",
          "resource_id": "1",
          "resource_type": "CustomButton",
          "created_at": "2015-09-10T12:55:40Z",
          "updated_at": "2015-09-16T14:28:17Z",
          "ae_namespace": "SYSTEM",
          "ae_class": "PROCESS",
          "ae_instance": "Request",
          "ae_attributes": {
            "key1": "val1",
            "key2": "val2",
            "request": "InspectMe"
          }
        }
      },
      {
        "id": "2",
        "guid": "7c900d18-57c1-11e5-8d22-000a27020067",
        "description": "Ping Service Status",
        "applies_to_class": "ServiceTemplate",
        "options": {
          "button_image": 3,
          "display": true
        },
        "userid": "admin",
        "created_on": "2015-09-10T13:40:23Z",
        "updated_on": "2015-09-16T14:16:21Z",
        "name": "Ping",
        "visibility": {
          "roles": [
            "_ALL_"
          ]
```

```
      },
      "applies_to_id": "7",
      "resource_action": {
        "id": "8",
        "resource_id": "2",
        "resource_type": "CustomButton",
        "created_at": "2015-09-10T13:40:23Z",
        "updated_at": "2015-09-16T14:16:21Z",
        "ae_namespace": "SYSTEM",
        "ae_class": "PROCESS",
        "ae_instance": "Request",
        "ae_attributes": {
          "Class": "System",
          "rel2": "//ManageIQ/System/Request/InspectMe",
          "request": "InspectMe"
        }
      }
    }
  ],
  "button_groups": [
    {
      "id": "43",
      "name": "MyButtonGroup|ServiceTemplate|7",
      "description": "My Custom Buttons",
      "created_on": "2015-09-10T13:41:20Z",
      "updated_on": "2015-09-15T14:53:49Z",
      "guid": "9e4a9900-57c1-11e5-8d22-000a27020067",
      "set_data": {
        "button_order": [
          3
        ],
        "button_image": 1,
        "display": true,
        "applies_to_class": "ServiceTemplate",
        "applies_to_id": "7",
        "group_index": 1
      },
      "owner_type": "ServiceTemplate",
      "owner_id": "7",
      "buttons": [
        {
          "id": "3",
          "guid": "92ac7d88-5bb9-11e5-b2f7-000a27020067",
          "description": "Simulate Button Dialog",
          "applies_to_class": "ServiceTemplate",
          "options": {
            "button_image": 12,
            "display": true
          },
          "userid": "admin",
          "created_on": "2015-09-15T14:53:49Z",
          "updated_on": "2015-09-16T14:28:50Z",
          "name": "simulate",
          "visibility": {
            "roles": [
              "_ALL_"
            ]
          },
          "applies_to_id": "7",
          "resource_action": {
            "id": "12",
            "dialog_id": "2",
            "resource_id": "3",
            "resource_type": "CustomButton",
            "created_at": "2015-09-15T14:53:49Z",
            "updated_at": "2015-09-16T14:28:50Z",
            "ae_namespace": "SYSTEM",
            "ae_class": "PROCESS",
            "ae_instance": "Request",
            "ae_attributes": {
              "request": "InspectMe"
            },
            "dialog": {
              "description": "Test Dialog 2",
              "buttons": "submit,cancel",
              "label": "TestDialog2",
              "dialog_tabs": [
                {
                  "description": "VM Sizing",
                  "display": "edit",
                  "label": "VmSizing",
```

```
                  "position": 0,
                  "dialog_groups": [
                    {
                      "description": "Memory",
                      "display": "edit",
                      "label": "Memory",
                      "position": 0,
                      "dialog_fields": [
                        {
                          "name": "DynamicMemory",
                          "description": "Amount of Memory for VM is Dynamic",
                          "type": "DialogFieldCheckBox",
                          "display": "edit",
                          "display_method_options": {
                          },
                          "required_method_options": {
                          },
                          "default_value": "f",
                          "values_method_options": {
                          },
                          "options": {
                          },
                          "label": "DynamicMemory",
                          "position": 0,
                          "dynamic": false,
                          "read_only": false,
                          "resource_action": {
                            "resource_type": "DialogField",
                            "ae_attributes": {
                            }
                          }
                        }
                      ]
                    }
                  ]
                }
              ]
            }
          }
        }
      ]
    }
  ]
},
"custom_action_buttons": [
  {
    "id": "1",
    "guid": "3d5975fe-57bb-11e5-8d22-000a27020067",
    "description": "Rescan Service Offering",
    "applies_to_class": "ServiceTemplate",
    "options": {
      "button_image": 2,
      "display": true
    },
    "userid": "admin",
    "created_on": "2015-09-10T12:55:40Z",
    "updated_on": "2015-09-16T14:28:17Z",
    "name": "Rescan",
    "visibility": {
      "roles": [
        "_ALL_"
      ]
    },
    "applies_to_id": "7"
  },
  {
    "id": "2",
    "guid": "7c900d18-57c1-11e5-8d22-000a27020067",
    "description": "Ping Service Status",
    "applies_to_class": "ServiceTemplate",
    "options": {
      "button_image": 3,
      "display": true
    },
    "userid": "admin",
    "created_on": "2015-09-10T13:40:23Z",
    "updated_on": "2015-09-16T14:16:21Z",
    "name": "Ping",
    "visibility": {
      "roles": [
        "_ALL_"
```

```
        ]
      },
      "applies_to_id": "7"
    },
    {
      "id": "3",
      "guid": "b9afcdfa-57c1-11e5-8d22-000a27020067",
      "description": "Simulate Button Dialog",
      "applies_to_class": "ServiceTemplate",
      "options": {
        "button_image": 3,
        "display": true
      },
      "userid": "admin",
      "created_on": "2015-09-10T13:42:06Z",
      "updated_on": "2015-09-16T14:27:30Z",
      "name": "Simulate",
      "visibility": {
        "roles": [
          "_ALL_"
        ]
      },
      "applies_to_id": "7"
    }
  }
  "actions": [
    {
      "name": "edit",
      "method": "post",
      "href": "http://localhost:3000/api/service_templates/7"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "http://localhost:3000/api/service_templates/7"
    }
  ]
}
```

When service templates define custom buttons, the ordered services are extended with custom actions which can be triggered via a POST action.

Note that the custom actions are exposed as actions on the deployed services, not the service_templates. The action name to trigger the action is the lower-cased name of the custom button.

As an example, here's a service without custom buttons:

```
{
  "href": "http://localhost:3000/api/services/91",
  "id": "91",
  "name": "new_service_91",
  "description": "This is a description for the new service 91",
  ...
  "actions": [
    {
      "name": "edit",
      "method": "post",
      "href": "http://localhost:3000/api/services/91"
    },
    {
      "name": "request_retire",
      "method": "post",
      "href": "http://localhost:3000/api/services/91"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "http://localhost:3000/api/services/91"
    }
  ]
}
```

Here is an example of a service with custom buttons:

```
{
  "href": "http://localhost:3000/api/services/92",
  "id": "92",
  "name": "enhanced_service_92",
  "description": "This is a description for the enhanced service 92",
  "guid": "24c193a2-57c8-11e5-9332-000a27020067",
  "service_template_id": "7",
  "options": {
```

```
    "button_order": [
      "cb-1",
      "cb-2",
      "cb-3"
    ],
    "dialog": {
      "dialog_DynamicMemory": "t"
    }
  },
  "display": true,
  "created_at": "2015-09-10T14:28:02Z",
  "updated_at": "2015-09-10T14:28:02Z",
  "evm_owner_id": "1",
  "miq_group_id": "1",
  "actions": [
    {
      "name": "edit",
      "method": "post",
      "href": "http://localhost:3000/api/services/92"
    },
    {
      "name": "request_retire",
      "method": "post",
      "href": "http://localhost:3000/api/services/92"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "http://localhost:3000/api/services/92"
    },
    {
      "name": "rescan",
      "method": "post",
      "href": "http://localhost:3000/api/services/92"
    },
    {
      "name": "ping",
      "method": "post",
      "href": "http://localhost:3000/api/services/92"
    },
    {
      "name": "simulate",
      "method": "post",
      "href": "http://localhost:3000/api/services/92"
    }
  ]
}
```

Here, the *rescan*, *ping* and *simulate* are custom buttons for the service and are returned after the common actions.

For example, to request the *rescan* custom action shown above, this is simply a POST action as follows:

Request:

```
POST /api/services/92

{
  "action" : "rescan"
}
```

Response:

Responses from custom action are the normal action responses from the REST API:

```
{
  "success": true,
  "message": "Invoked custom action rescan for services id: 92",
  "href": "http://localhost:3000/api/services/92"
}
```

- Service Dialogs
  - Creating Service Dialogs
  - Editing Service Dialogs
  - Copying Service Dialogs
  - Refreshing Dialog Fields of Service Dialogs
  - Deleting Service Dialogs

Service Dialogs

Management of Service Dialogs is provided via the collection:

`/api/service_dialogs`

The following actions are provided on service_dialogs resources:

- Creating Service Dialogs
- Editing Service Dialogs
- Copying Service Dialogs
- Refreshing Dialog Fields of Service Dialogs
- Deleting Service Dialogs

Creating Service Dialogs

Service Dialogs can be created via a POST to the *service_dialogs* collection or via the **create** action signature which also allows creation of multiple service_dialogs in a single request.

```
POST /api/service_dialogs
```

```
{
  "description" : "Dialog",
  "label" : "dialog_label",
  "dialog_tabs" : [
    {
      "description" : "Dialog Tab",
      "position"    : 0,
      "label"       : "dialog_tab_label",
      "dialog_groups" : [
        {
          "description" : "Dialog Group",
          "label"       : "group_label",
          "dialog_fields" : [
            {
              "name"  : "A Dialog Field",
              "label" : "dialog_field_label"
            }
          ]
        }
      ]
    }
  ]
}
```

or creating multiple Service Dialogs:

```
{
  "action" : "create",
  "resources" : [
    { "description" : "Dialog 1", "label" : "dialog_1_label", ... },
    { "description" : "Dialog 2", "label" : "dialog_2_label", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Service Dialogs.

Editing Service Dialogs

Service Dialogs can be updated via the *edit* action as follows:

```
POST /api/service_dialogs/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "label" : "updated_label",
    "content" : {
      "dialog_tabs" : [
        "id" : "11",
        "label" : "updated_tab_label",
        "dialogs_groups" : [
```

```
          {
            "id" : "31",
            "dialog_fields" : [
               { "id" : "41" }
            ]
          }
        ]
      ]
    }
  }
}
```

or editing multiple Service Dialogs in a single request as follows:

```
POST /api/service_dialogs
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/service_dialogs/21",
      "label" : "updated_label_1",
      "content" : ...
    },
    {
      "href" : "http://localhost:3000/api/service_dialogs/22",
      "label" : "updated_label_2",
      "content" : ...
    },
    ...
  ]
}
```

Copying Service Dialogs

Service Dialogs can be copied to create new Service Dialogs via the *copy* action as follows:

```
POST /api/service_dialogs/:id
```

```
{
  "action" : "copy"
}
```

When a label is not specified, the new Service Dialog will be named **Copy of ...** the name of the Service Dialog being copied

Optionally, the label of the new Service Dialog can also be specified as follows:

```
{
  "action" : "copy"
  "resource" : {
    "label" : "test_service_dialog_label"
  }
}
```

Multiple Service Dialogs can also be copied in a single request as follows:

```
POST /api/service_dialogs
```

```
{
  "action" : "copy",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/service_dialogs/11",
      "label" : "test_service_dialog_label_1"
    },
    {
      "href" : "http://localhost:3000/api/service_dialogs/12",
      "label" : "test_service_dialog_label_2"
    },
    ...
  ]
}
```

Refreshing Dialog Fields of Service Dialogs

Refreshing dialog fields for a Service Dialog is done as follows:

```
POST /api/service_dialogs/:id
```

```
{
  "action" : "refresh_dialog_fields",
  "resource" : {
```

```
    "resource_action_id" : "6",
    "target_id" : "12",
    "target_type" : "service_template_ansible_tower",
    "dialog_fields" : {
      "my_field1" : "my field1 value",
      "my_field2" : "my field2 value",
      "my_field3" : "my field3 value",
      ...
    },
    "fields" : [ "my_field1", "my_field3" ]
  }
}
```

Deleting Service Dialogs

Deleting a Service Dialog can be done via the *delete* action as follows:

```
POST /api/service_dialogs/:id
```

```
{
  "action" : "delete"
}
```

or via the DELETE HTTP method as follows:

```
DELETE /api/service_dialogs/:id
```

One can also delete multiple Service Dialogs via the *delete* action on the collection with references to the service dialogs to delete as follows:

```
POST /api/service_dialogs
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_dialogs/101" },
    { "href" : "http://localhost:3000/api/service_dialogs/102" },
    { "href" : "http://localhost:3000/api/service_dialogs/103" },
    ...
  ]
}
```


- Service Orders
  - Querying Service Orders
  - Creating Service Orders with multiple service requests
  - Adding Service Requests
  - Removing Service Requests
  - Clearing Service Orders
  - Copying Service Orders
  - Deleting Service Orders
  - Ordering Service Orders

Service Orders

The service orders collection and actions are provided for managing Shopping Carts and their related service requests.

```
/api/service_orders
```

Available actions on service_orders instances include:

Action	Description
create	Creating the Shopping Cart
clear	Clearing the Shopping Cart
order	Ordering all service requests in the Shopping Cart
copy	Copying the Shopping Cart
delete	Deleting the Shopping Cart

Some important points on Shopping Carts:

- There is one and only one shopping cart for the authenticated user
- A shopping cart is an /api/service_orders resource with a state of **cart**
- Shopping carts can be referenced by their resource id or preferably via the alphanumeric **cart** identifier.
- resource id's need to be used for accessing non-cart service_orders states of **wish** list and **ordered**

Items to add to a shopping cart are *service_requests* that are specified similarly to when services are ordered from a service template.

- Querying Service Orders
- Creating Service Orders
- Adding Service Requests
- Removing Service Requests
- Clearing Service Orders
- Copying Service Orders
- Deleting Service Orders
- Ordering Service Orders

Querying Service Orders

All the queries listed here below return the service orders for the authenticated user.

Getting all service_orders:

```
GET /api/service_orders
```

Getting the shopping cart:

```
GET /api/service_orders/cart
```

Getting the shopping cart with expanded service_requests:

```
GET /api/service_orders/cart?expand=service_requests,resources
```

Getting the service requests of the shopping cart:

```
GET /api/service_orders/cart/service_requests
```

Creating Service Orders with multiple service requests

```
POST /api/service_orders
```

```
{
  "service_requests" : [
    {
      "service_template_href" : "http://localhost:3000/api/service_templates/3",
      "Basic_Host_Name" : "super_vm_a",
      "Basic_Host_Ip" : "192.168.120.101"
    },
    {
      "service_template_href" : "http://localhost:3000/api/service_templates/3",
      "Basic_Host_Name" : "super_vm_b",
      "Basic_Host_Ip" : "192.168.120.102"
    },
    ...
  ]
}
```

Creating a wish list service_order:

```
POST /api/service_orders
```

```
{
  "state" : "wish",
  "service_requests" : [
    {
      "service_template_href" : "http://localhost:3000/api/service_templates/2",
      "host_name" : "sample_service_order_host_name",
      "host_ip" : "192.168.120.101"
    }
  ]
}
```

Adding Service Requests

Adding service requests to a shopping cart can be done by either of the create methods shown above which will all add the service requests to the existing shopping cart:

```
POST /api/service_orders
```

or specifically to the shopping cart or any wish list service_orders via its *id* instead of **cart** as follows:

```
POST /api/service_orders/cart/service_requests

POST /api/service_orders/:id/service_requests

{
  "action" : "add",
  "resources" : [
    {
      "service_template_href" : "http://localhost:3000/api/service_templates/3",
      "Basic_Host_Name" : "sample_service_order_host_name_2",
      "Basic_Host_Ip" : "192.168.120.102"
    },
    {
      "service_template_href" : "http://localhost:3000/api/service_templates/3",
      "Basic_Host_Name" : "sample_service_order_host_name_3",
      "Basic_Host_Bad_Ip" : "192.168.120.103"
    },
    ...
  ]
}
```

Note: * POST /api/service_orders is atomic, requests are validated for all requests before adding them. * POST /api/service_orders/cart/service_requests will add the requests one by one, the action responses have the **success** field for each requests reflecting the success or failure of each request.

Removing Service Requests

Removing requests from the shopping cart is available via the *remove* action:

```
POST /api/service_orders/cart/service_requests

{
  "action" : "remove",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_orders/6/service_requests/56" },
    { "href" : "http://localhost:3000/api/service_orders/6/service_requests/57" }
  ]
}
```

Clearing Service Orders

Clearing the shopping cart can be done via the *clear* action:

```
POST /api/service_orders/cart

{
  "action" : "clear"
}
```

Copying Service Orders

A service order can be copied into a new service order via the *copy* action:

```
POST /api/service_orders/:id

{
   "action" : "copy",
   "resource" : {
      "name" : "new_service_order_name"
   }
}
```

Service orders can also be copied in bulk as follows:

```
POST /api/service_orders/:id

{
   "action" : "copy",
   "resources" : [
      {
        "href" : "http://localhost:3000/api/service_orders/1",
        "name" : "new_service_order_name_1"
      },
      {
        "href" : "http://localhost:3000/api/service_orders/2",
        "name" : "new_service_order_name_2"
      },
      ...
```

```
   ]
}
```

Deleting Service Orders

Deleting the shopping cart can be done via the *delete* action:

```
POST /api/service_orders/cart
```

```
{
   "action" : "delete"
}
```

or simply via the DELETE method:

```
DELETE /api/service_orders/cart
```

Ordering Service Orders

Ordering the shopping cart is done via the *order* action as follows:

```
POST /api/service_orders/cart
```

```
{
   "action" : "order"
}
```


- Setting Ownership
  - Specifying Ownership
  - Setting Ownership to single resource
    - Request:
    - Response:
  - Setting Ownership to multiple resources
    - Request:
    - Response:

Setting Ownership

Setting ownership of resources is provided with the **set_ownership** action. This action is available for the following resources:

/api/auth_key_pairs
/api/cloud_templates
/api/instances
/api/service_templates
/api/services
/api/templates
/api/vms

- Specifying Ownership
- Setting Ownership to single resource
- Setting Ownership to multiple resources

Specifying Ownership

Specifying ownership in requests is done via the **owner** and **group** specification.

```
{
  "owner" : { "href" : "http://localhost:3000/api/users/:id"  },
  "group" : { "href" : "http://localhost:3000/api/groups/:id" }
}
```

Note:

Specifying an owner can be done via **href**, **id**, **name** or **userid**

Note:

Specifying a group can be done via **href**, **id** or **description**

Setting ownership can be done on a specific resource or multiple resources in a single request:

Setting Ownership to single resource

Example setting ownership to a single resource:

Request:

```
POST /api/vms/320
```

```
{
  "action" : "set_ownership",
  "resource" : {
    "owner" : { "userid" : "jdoe" },
    "group" : { "description" : "TestGroup" }
  }
}
```

Response:

```
{
  "success": true,
  "message": "setting ownership of vms id 320 to owner: jdoe, group: TestGroup",
  "href": "http://localhost:3000/api/vms/320"
}
```

Setting Ownership to multiple resources

Example setting ownership to multiple resources with a single request:

Request:

```
POST /api/services
```

```
{
  "action" : "set_ownership",
  "resources" : [
    { "href" : "http://localhost:3000/api/services/104", "owner" : { "name" : "John Doe" } },
    { "href" : "http://localhost:3000/api/services/105", "owner" : { "name" : "John Doe" } },
    { "href" : "http://localhost:3000/api/services/106", "owner" : { "name" : "John Doe" } }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "setting ownership of services id 104 to owner: John Doe",
      "href": "http://localhost:3000/api/services/104"
    },
    {
      "success": true,
      "message": "setting ownership of services id 105 to owner: John Doe",
      "href": "http://localhost:3000/api/services/105"
    },
    {
      "success": true,
      "message": "setting ownership of services id 106 to owner: John Doe",
      "href": "http://localhost:3000/api/services/106"
    }
}
```


- Settings Management
  - Querying Settings
  - Updating Settings
    - Request:
    - Response:
  - Deleting Settings
    - Example:

Settings Management

Management of advanced settings is available on servers, zones and regions. They are provided with the following collection:

```
/api/settings
```

for the server processing the API, or for a particular server, zone or region as follows:

```
/api/servers/:id/settings
```

```
/api/zones/:id/settings
```

```
/api/regions/:id/settings
```

The following actions are available with advanced settings:

- Querying Settings
- Updating Settings
- Deleting Settings

Querying Settings

Settings can be queried as follows:

```
GET /api/servers
```

```
GET /api/servers/:id/settings
```

```
GET /api/zones/:id/settings
```

```
GET /api/regions/:id/settings
```

Note:

Authorization

- For users with an *EvmGroup-super_administrator* role, the complete advanced settings are returned.
- For users with roles that include the *Configuration Settings* feature, only a whitelisted set of settings are returned.
- Requests from all other users will be rejected with a Forbidden error.

Updating Settings

Updating settings can only be performed by users with an *EvmGroup-super_administrator* role.

Updates to settings is available with the HTTP PATCH method as follows:

```
PATCH /api/servers/:id/settings
```

```
PATCH /api/zones/:id/settings
```

```
PATCH /api/regions/:id/settings
```

The following is an example updating the policy_events's purge_window_size:

Request:

```
PATCH /api/servers/1/settings
```

```
{
  "policy_events" : {
    "purge_window_size" : 500
  }
}
```

Response:

The complete updated settings is returned:

```
{
  "ems": {
    "ems_ansible_tower": {
      "blacklisted_event_names": [

      ],
  ...
  "policy_events" : {
    "history": {
      "keep_policy_events": "6.months",
      "purge_window_size": 1000
    },
    "purge_window_size" : 500
  },
```

```
  ...
}
```

Deleting Settings

Deleting settings can only be performed by users with an *EvmGroup-super_administrator* role.

Deleting settings is available with the HTTP DELETE method as follows:

```
DELETE /api/servers/:id/settings
```

```
DELETE /api/zones/:id/settings
```

```
DELETE /api/regions/:id/settings
```

Requests must include an array representing the path of the setting to delete.

Example:

```
DELETE /api/servers/1/settings
```

```
[ "policy_events", "purge_window_size" ]
```

This request deletes the policy_event's purge_window_size setting for /api/servers/1 and its value would revert to its default.

- Snapshot Management
  - Snapshot Actions
  - Querying Snapshots
  - Creating Snapshots
  - Reverting Snapshots
  - Deleting Snapshots

Snapshot Management

Management of snapshots is provided for virtual machines and instances via the snapshots subcollection:

```
/api/vms/:id/snapshots
```

```
/api/instances/:id/snapshots
```

Snapshot Actions

The following snapshot actions are available for the supported resources:

Snapshot Action	Vms	Instances
read	X	X
create	X	X
revert	X	
delete	X	X

- Querying Snapshots
- Creating Snapshots
- Reverting Snapshots
- Deleting Snapshots

Querying Snapshots

Snapshots of machines are queried via the subcollection of vms and instances as follows:

Querying all snapshots of a VM:

```
GET /api/vms/:id/snapshots
```

Querying all snapshots of an Instance:

```
GET /api/instances/:id/snapshots
```

Filtering, sorting and paging on the subcollection as mentioned on the Querying page.

For example:

```
GET /api/vms/11/snapshots?expand=resources&filter[]=name=test*
```

One can also fetch a single snapshot by id:

```
GET /api/vms/:id/snapshots/:snapshot_id
```

```
GET /api/instances/:id/snapshots/:snapshot_id
```

Creating Snapshots

Creating snapshots on Vms and Instances is done by posting the request to the snapshots subcollection.

```
POST /api/vms/:id/snapshots
```

```
POST /api/instances/:id/snapshots
```

```
{
  "name" : "name_of_new_snapshot",
  "description" : "Description of new Snapshot",
  "memory" : false
}
```

In the above example, only *name* is required. *memory* can be set to true to have the snapshot include physical memory.

Reverting Snapshots

Reverting snapshots is only available for Vms.

```
POST /api/vms/:id/snapshots/:snapshot_id
```

```
{
  "action" : "revert"
}
```

Deleting Snapshots

Individual snapshots of Vms and Instances can be deleted via the DELETE method as follows:

```
DELETE /api/vms/:id/snapshots/:snapshot_id
```

```
DELETE /api/instances/:id/snapshots/:snapshot_id
```

As well as the *delete* POST action as follows for VMs:

```
POST /api/vms/:id/snapshots/:snapshot_id
```

or instances:

```
POST /api/instances/:id/snapshots/:snapshot_id
```

```
{
  "action" : "delete"
}
```

Snapshots can also be deleted in bulk via the *delete* action on the snapshots subcollection as follows:

```
POST /api/vms/11/snapshots
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/vms/11/snapshots/3" },
    { "href" : "http://localhost:3000/api/vms/11/snapshots/4" },
    { "href" : "http://localhost:3000/api/vms/11/snapshots/5" }
  ]
}
```

As well as for instances:

```
POST /api/instances/201/snapshots
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/instances/201/snapshots/7" },
    { "href" : "http://localhost:3000/api/instances/201/snapshots/8" }
  ]
}
```

- Tags & Tagging
  - Tag Management
  - Querying Tags
  - Querying Tags of a Resource

- Assigning Tags
  - Tag specification
  - Assigning tags to a resource
    - Response:
  - Unassigning tags from a resource
    - Response:
  - Bulk Assigning Tags
    - Response:
  - Bulk Unassigning Tags
    - Response:

Tags & Tagging

- Tag Management
- Querying Tags
- Querying Tags of a Resource
- Assigning Tags
- Tag Specification
- Assigning Tags to a Resource
- Unassigning Tags from a Resource
- Bulk Assigning Tags
- Bulk Unassigning Tags

Tag Management

Management of Tags is provided in the following forms:

- Querying of tags via the primary collection:

`/api/tags`

- Assigning and Unassigning tags to resources via the tags subcollection of a resource

`/api/:collection/:id/tags`

and posting **assign** or **unassign** actions to the tags subcollection.

Querying Tags

Querying all tags in the system is simply:

`GET /api/tags`

However, that only gives us the href's of the tags:

```
{
  "name": "tags",
  "count": 1385,
  "subcount": 1385,
  "resources": [
    {
      "href": "http://localhost:3000/api/tags/1"
    },
    {
      "href": "http://localhost:3000/api/tags/2"
    },
    {
      "href": "http://localhost:3000/api/tags/3"
    }
    ...
  ]
}
```

Which is not too useful.

Expanding the resources, does provide us with the tag name as follows:

`GET /api/tags?expand=resources`

```
{
  "name": "tags",
  "count": 1385,
  "subcount": 1385,
  "resources": [
    {
      "href": "http://localhost:3000/api/tags/1",
      "id": "1",
      "name": "/managed/roles/change_managers"
    },
    {
      "href": "http://localhost:3000/api/tags/2",
      "id": "2",
      "name": "/managed/roles/operators"
    },
    {
      "href": "http://localhost:3000/api/tags/3",
      "id": "3",
      "name": "/managed/roles/cfme_administrators"
    },
    {
      "href": "http://localhost:3000/api/tags/4",
      "id": "4",
      "name": "/managed/location"
    },
    {
      "href": "http://localhost:3000/api/tags/5",
      "id": "5",
      "name": "/managed/location/ny"
    }
    ...
  ]
}
```

For additional details on tags, the **category** and **classification** attributes can be queried on tags as follows:

```
GET /api/tags/5?attributes=category,classification
```

Additionally, a compound attribute called **categorization** is provided to return key information from both **category** and **classification** of tags as follows:

```
GET /api/tags/5?attributes=categorization
```

```
{
  "href": "http://localhost:3000/api/tags/5",
  "id": "5",
  "name": "/managed/location/ny",
  "categorization": {
    "name": "ny",
    "description": "New York",
    "category": {
      "name": "location",
      "description": "Location"
    },
    "display_name": "Location: New York"
  }
}
```

Querying Tags of a Resource

To find out which tags are assigned to a resource, the tags subcollection can be expanded for that particular resource as follows:

```
GET /api/vms/320?expand=tags
```

This will provide the id, href and name of the tags:

```
{
  "href": "http://localhost:3000/api/vms/320",
  "id": "320",
  "vendor": "vmware",
  "name": "aab-vm1",
  "description": "this is a test",
  "raw_power_state": "poweredOn",
  ...
  "tags": [
    {
      "href": "http://localhost:3000/api/vms/320/tags/81",
      "id": "81",
      "name": "/managed/quota_max_cpu/4"
    },
    {
```

```
      "href": "http://localhost:3000/api/vms/320/tags/61",
      "id": "61",
      "name": "/managed/cc/001"
    },
    ...
  ]
}
```

Optionally, one can also query just the tags subcollection of the resource to get additional information about them like the categorization:

```
GET /api/vms/320/tags?expand=resources&attributes=categorization
```

```
{
  "name": "tags",
  "count": 1385,
  "subcount": 5,
  "resources": [
    {
      "href": "http://localhost:3000/api/vms/320/tags/81",
      "id": "81",
      "name": "/managed/quota_max_cpu/4",
      "categorization": {
        "name": "4",
        "description": "4",
        "category": {
          "name": "quota_max_cpu",
          "description": "Quota - Max CPUs"
        },
        "display_name": "Quota - Max CPUs: 4"
      }
    },
    {
      "href": "http://localhost:3000/api/vms/320/tags/61",
      "id": "61",
      "name": "/managed/cc/001",
      "categorization": {
        "name": "001",
        "description": "Cost Center 001",
        "category": {
          "name": "cc",
          "description": "Cost Center"
        },
        "display_name": "Cost Center: Cost Center 001"
      }
    },
    ...
  ]
}
```

Assigning Tags

Tag management on resources can be done by POSTing **assign** and **unassign** actions to the tags subcollection of resources. Tags can be managed as subcollections of the following primary collections:

Collection
/api/categories
/api/clusters
/api/data_stores
/api/generic_objects
/api/groups
/api/hosts
/api/lans
/api/providers
/api/resource_pools
/api/services
/api/service_templates
/api/templates
/api/tenants
/api/users
/api/vms

Tag specification

Tags can be specified using one of the following forms:

By Tag category and name:

```
{
  "category" : "department",
  "name"     : "finance"
}
```

By Tag path:

```
{
  "path" : "/managed/department/finance"
}
```

By Tag href:

```
{
  "href" : "http://localhost:3000/api/tags/10"
}
```

Assigning tags to a resource

```
POST /api/vms/320/tags
```

```
{
  "action" : "assign",
  "resources" : [
    { "category" : "department", "name" : "finance" },
    { "category" : "cc", "name" : "001" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/vms/320",
      "tag_category": "department",
      "tag_name": "finance",
      "tag_href": "http://localhost:3000/api/tags/81"
    },
    {
      "success": true,
      "message": "Assigning Tag: category:'cc' name:'001'",
      "href": "http://localhost:3000/api/vms/320",
      "tag_category": "cc",
      "tag_name": "001",
      "tag_href": "http://localhost:3000/api/tags/91"
    }
}
```

Unassigning tags from a resource

```
POST /api/vms/320/tags
```

```
{
  "action" : "unassign",
  "resources" : [
    { "category" : "department", "name" : "finance" },
    { "category" : "cc", "name" : "001" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Unassigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/vms/320",
      "tag_category": "department",
      "tag_name": "finance",
      "tag_href": "http://localhost:3000/api/tags/81"
    },
    {
      "success": true,
```

```
      "message": "Unassigning Tag: category:'cc' name:'001'",
      "href": "http://localhost:3000/api/vms/320",
      "tag_category": "cc",
      "tag_name": "001",
      "tag_href": "http://localhost:3000/api/tags/91"
    }
}
```

Additional Tag Management examples can be found on the main REST API Examples section.

Bulk Assigning Tags

While the general URL for tag assignment via the subcollection as follows:

```
POST /api/vms/320/tags
```

allows one to assign multiple tags to a single resource in one call, it does not provide the ability to assign tags to multiple resources in one call.

Bulk assignment of tags is available via the *assign_tags* action to the following collections:

Collection
/api/vms
/api/services

The *assign_tags* action allows adding one or more tags to one or more resources.

```
POST /api/vms
```

```
{
  "action" : "assign_tags",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/vms/11",
      "tags" : [
        { "category" : "department", "name" : "finance" }
      ]
    },
    {
      "href" : "http://localhost:3000/api/vms/12",
      "tags" : [
        { "category" : "cc", "name" : "001" }
      ]
    }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/vms/11",
      "tag_category": "department",
      "tag_name": "finance",
      "tag_href": "http://localhost:3000/api/tags/81"
    },
    {
      "success": true,
      "message": "Assigning Tag: category:'cc' name:'001'",
      "href": "http://localhost:3000/api/vms/12",
      "tag_category": "cc",
      "tag_name": "001",
      "tag_href": "http://localhost:3000/api/tags/91"
    }
  ]
}
```

Bulk Unassigning Tags

Tags can also be unassigned in bulk from multiple resources for the following collections via the *unassign_tags* action:

Collection
/api/vms
/api/services

```
POST /api/services
```

```
{
  "action" : "unassign_tags",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/services/5",
      "tags" : [
        { "href" : "http://localhost:3000/api/services/5/tags/81" }
      ]
    },
    {
      "href" : "http://localhost:3000/api/services/6",
      "tags" : [
        { "category" : "cc", "name" : "001" }
      ]
    }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Unassigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/services/5",
      "tag_category": "department",
      "tag_name": "finance",
      "tag_href": "http://localhost:3000/api/tags/81"
    },
    {
      "success": true,
      "message": "Unassigning Tag: category:'cc' name:'001'",
      "href": "http://localhost:3000/api/services/6",
      "tag_category": "cc",
      "tag_name": "001",
      "tag_href": "http://localhost:3000/api/tags/91"
    }
  ]
}
```


- Tasks
  - Querying Tasks
  - Deleting Tasks

Tasks

Management of Tasks is available via the following endpoint:

`/api/tasks`

Querying Tasks

Querying all Tasks:

`GET /api/tasks`

Getting details of a specific Task:

`GET /api/tasks/:id`

The following actions are available on tasks:

- Deleting Tasks

Deleting Tasks

Tasks can be deleted via either the **delete** POST action or the DELETE HTTP method.

`POST /api/tasks/101`

```
{
  "action" : "delete"
}
```

or simply:

`DELETE /api/tasks/101`

Deleting multiple Tasks can be done as follows:

```
POST /api/tasks

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/tasks/101" },
    { "href" : "http://localhost:3000/api/tasks/102" },
    ...
  ]
}
```


- Tenant Management
- Tenant Quota Management
  - Creating Tenants
  - Editing Tenants
  - Deleting Tenants
  - Creating Tenant Quotas
  - Editing Tenant Quotas
  - Deleting Tenant Quotas

Tenant Management

Tenant management including CRUD operations is available via the REST API. Management of tenants is through:

```
/api/tenants
```

- Creating Tenants
- Editing Tenants
- Deleting Tenants

Tenant Quota Management

Tenant Quota management is made available via the *quotas* subcollection of tenant resources as follows:

```
/api/tenants/:id/quotas
```

- Creating Tenant Quotas
- Editing Tenant Quotas
- Deleting Tenant Quotas

Creating Tenants

Creating a Tenant is done via the **create** action as follows:

```
POST /api/tenants

{
  "action" : "create",
  "resource" : {
    # New Tenant JSON
  }
}
```

And like other **create** actions, the new tenant data can simply be posted to the /api/tenants URL.

```
POST /api/tenants

{
  # New Tenant JSON
}
```

For example, creating a new tenant:

```
POST /api/tenants

{
  "name" : "Test Tenant",
  "description" : "Test Tenant Description",
  "parent" : { "href" : "http://localhost:3000/api/tenants/1" }
}
```

Parent can be specify by *id* or *href*

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Tenants.

Editing Tenants

Editing a Tenant is available via the **edit** action with the **resource** specifying the attributes to update for that tenant.

```
POST /api/tenants/:id
```

```
{
  "action" : "edit" ,
  "resource" : {
    "description" : "Updated Description for Tenant"
  }
}
```

Note:

type is a restricted attribute and cannot be edited.

Deleting Tenants

Deleting a Tenant is done via the delete method:

```
DELETE /api/tenants/:id
```

or via the **delete** action as follows:

```
POST /api/tenants/:id
```

```
{
  "action" : "delete"
}
```

or by href:

```
POST /api/tenants
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/tenants/:id" }
  ]
}
```

Creating Tenant Quotas

Creating a quota for a specific tenant is done as follows:

```
POST /api/tenants/:id/quotas
```

```
{
  "name" : "cpu_allocated",
  "value" : 1
}
```

Editing Tenant Quotas

```
POST /api/tenants/:id/quotas/:quota_id
```

```
{
  "action" : "edit",
  "resource" : {
    "value" : 5
  }
}
```

Multiple quotas for a tenant can also be edited in a single request as follows:

```
POST /api/tenants/1/quotas
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/tenants/1/quotas/5",
      "value" : 11
    },
```

```
    { "href" : "http://localhost:3000/api/tenants/1/quotas/6",
      "value" : 12
    },
    ...
  ]
}
```

Deleting Tenant Quotas

Quotas of a tenant can be deleted via the *delete* action as follows:

```
POST /api/tenants/:id/quotas/:quota_id
```

```
{
  "action" : "delete"
}
```

or via the DELETE HTTP method

```
DELETE /api/tenants/:id/quotas/:quota_id
```

Multiple quotas of a tenant can be deleted in a single request as shown in the following example:

```
POST /api/tenants/1/quotas
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/tenants/1/quotas/7" },
    { "href" : "http://localhost:3000/api/tenants/1/quotas/8" },
    ...
  ]
}
```


- User Management
  - Querying Users
  - Creating Users
  - Editing Users
  - Editing Authenticated User
  - Deleting Users

User Management

Management of Users is provided via the following collection:

```
/api/users
```

Full CRUD actions on users is available:

- Querying Users
- Creating Users
- Editing Users
- Editing Authenticated User
- Deleting Users

Querying Users

Querying all users in the system is simply:

```
GET /api/users
```

Getting details on a specific user:

```
GET /api/users/:id
```

Creating Users

Users can be created via a POST to the users collection or via the **create** action signature which also allows creation of multiple users in a single request.

```
POST /api/users
```

```
{
  "userid" : "testuser",
```

```
  "password" : "secret",
  "name" : "Test User",
  "group" : { "id" : "11" }
}
```

Note:

group can be specified via **description**, **href**, or **id**

or creating multiple users:

```
{
  "action" : "create",
  "resources" : [
    { "userid" : "testuser1", "name" : "Test User1", ... },
    { "userid" : "testuser2", "name" : "Test User2", ... },
    ...
  ]
}
```

Note:

Please refer to the Resource Attributes page for a list of available attributes when creating Users.

Editing Users

```
POST /api/users/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "group" : { "id" : "12" }
  }
}
```

or editing multiple users:

```
POST /api/users
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/users/101",
      "group" : { "href" : "http://localhost:3000/api/groups/15" }
    },
    {
      "href" : "http://localhost:3000/api/users/102",
      "group" : { "href" : "http://localhost:3000/api/groups/15" }
    },
    {
      "href" : "http://localhost:3000/api/users/103",
      "group" : { "href" : "http://localhost:3000/api/groups/15" }
    },
    ...
  ]
}
```

Editing Authenticated User

An API Authenticated User can edit some of their own attributes without having general user edit permissions. Editable attributes for the authenticated user are:

password
email
settings

Users can also update their currently selected group with the **set_current_group** action as follows:

```
POST /api/users/:id
```

```
{
  "action" : "set_current_group",
  "resource" : {
    "group" : { "description" : "EvmGroup-auditor" }
  }
}
```

Note:

The group reference can be via **description**, **href** or **id**. Also, only a group that the user belongs to can be specified.

Deleting Users

users can be deleted via either the **delete** POST action or via the DELETE HTTP method.

```
POST /api/users/101

{
  "action" : "delete"
}
```

or simply:

```
DELETE /api/users/101
```

Deleting multiple users can be done as follows:

```
POST /api/users

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/users/101" },
    { "href" : "http://localhost:3000/api/users/102" },
    ...
  ]
}
```


- Virtual Machine Management
  - Targeting Virtual Machines
  - Querying Virtual Machines
  - Starting a Virtual Machine
  - Stopping a Virtual Machine
  - Suspending a Virtual Machine
  - Refreshing a Virtual Machine
  - Retiring a Virtual Machine
  - Resetting a Virtual Machine
  - Rebooting guest in Virtual Machine
  - Shutting Down Guest in a Virtual Machine
  - Scanning a Virtual Machine
  - Setting Server of a Virtual Machine
  - Setting Owner of a Virtual Machine
  - Adding an Event to a Virtual Machine
  - Editing a Virtual Machine
  - Deleting a Virtual Machine

Virtual Machine Management

Management of virtual machines adds support of the following actions:

Action	Description
start	Starts a virtual machine
stop	Stops a virtual machine
suspend	Suspends a virtual machine
refresh	Refreshes a virtual machine
request_retire	Retires a virtual machine
reset	Resets a virtual machine
reboot_guest	Reboots guest in virtual machine
shutdown_guest	Shuts down guest in virtual machine
scan	Scans a virtual machine (Perform SmartState Analysis)
set_miq_server	Sets the server of a virtual machine
set_owner	Sets the owner of a virtual machine
add_event	Adding an event to a virtual machine
edit	Edits a virtual machine
delete	Deletes a virtual machine in the appliance

- Targeting VMs

- Querying VMs
- Starting a VM
- Stopping a VM
- Suspending a VM
- Refreshing a VM
- Retiring a VM
- Resets a VM
- Reboots guest in VM
- Shuts down guest in VM
- Scanning a VM
- Setting Server of a VM
- Setting Owner of a VM
- Adding an Event to a VM
- Editing a VM
- Deleting a VM

Targeting Virtual Machines

These actions can be triggered on individual virtual machine resources:

```
/api/vms/:id
```

As simply as POSTing the following action to a virtual machine.

```
{
  "action" : "start"
}
```

Requests can also be made on multiple vms by targetting the primary collection:

```
/api/vms
```

```
{
  "action" : "start",
  "resources" : [
    { "href" : "http://localhost:3000/api/vms/11" },
    { "href" : "http://localhost:3000/api/vms/12" },
    ...
  ]
}
```

Querying Virtual Machines

Virtual machines are queried via the primary collection URL:

```
/api/vms
```

Filtering, sorting and paging as mentioned on the Querying page.

When querying vms, expanding the resources themselves as well as the following subcollections:

accounts
software

For example:

```
GET /api/vms?expand=resources,accounts,software
```

Or querying an individual virtual machine

```
GET /api/vms/:id?expand=accounts,software
```

Starting a Virtual Machine

```
{
  "action" : "start"
}
```

Stopping a Virtual Machine

```
{
  "action" : "stop"
}
```

Suspending a Virtual Machine

```
{
  "action" : "suspend"
}
```

Refreshing a Virtual Machine

```
{
  "action" : "refresh"
}
```

Retiring a Virtual Machine

```
{
  "action" : "request_retire"
}
```

Resetting a Virtual Machine

```
{
  "action" : "reset"
}
```

Rebooting guest in Virtual Machine

```
{
  "action" : "reboot_guest"
}
```

Shutting Down Guest in a Virtual Machine

```
{
  "action" : "shutdown_guest"
}
```

Scanning a Virtual Machine

```
{
  "action" : "scan"
}
```

Setting Server of a Virtual Machine

```
{
  "action" : "set_miq_server",
  "resource" : {
    "miq_server" : { "href" : "http://localhost:3000/api/servers/5" }
  }
}
```

```
{
  "action" : "set_miq_server",
  "resource" : {
    "miq_server" : { "id" : "6" }
  }
}
```

To remove the server from the virtual machine, pass in an empty reference as follows:

```
{
  "action" : "set_miq_server",
  "resource" : {
    "miq_server" : {}
  }
}
```

Setting Owner of a Virtual Machine

```
{
  "action" : "set_owner",
  "resource" : {
    "owner" : "admin"
  }
}
```

Adding an Event to a Virtual Machine

```
{
  "action" : "add_event",
  "resource" : {
    "event_type" : "...",
    "event_message" : "...",
    "event_time" : "UTC Time"
  }
}
```

Note:

event_time above is optional. If skipped, current time will be used.

Editing a Virtual Machine

Basic information of virtual machines can be edited. This includes the following:

VM Info	Attribute
description	description
custom attributes	custom_1 ... custom_9
parent resource	parent_resource - resource href reference
child resources	child_resources - array of resource href references

Virtual machine resources can be edited as follows:

```
POST /api/vms/:id
```

```
{
  "action" : "edit",
  "resource" : {
    "description" : "Updated VM Description",
    "custom_1" : "custom_attribute_1",
    "parent_resource" : { "href" : "http://localhost:3000/api/vms/11" },
    "child_resources" : [
      { "href" : "http://localhost:3000/api/vms/101" },
      { "href" : "http://localhost:3000/api/vms/102" },
      { "href" : "http://localhost:3000/api/vms/103" },
      { "href" : "http://localhost:3000/api/vms/104" }
    ]
  }
}
```

Virtual machines can also be edited in bulk as follows:

```
POST /api/vms
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/vms/11",
      "custom_9" : "vm_class_a"
    },
    {
      "href" : "http://localhost:3000/api/vms/12",
      "custom_9" : "vm_class_a"
    },
    {
      "href" : "http://localhost:3000/api/vms/13",
      "custom_9" : "vm_class_a"
    }
  ]
}
```

Deleting a Virtual Machine

```
{
  "action" : "delete"
}
```

Or simply doing the following:

```
DELETE /api/vms/:id
```

See the main REST API examples section for additional virtual machine management examples.

Infrastructure management API Appendices

In this guide you will find appendices for the Infrastructure Automation – Infrastructure Management REST API.

- Quick Reference
- Collection Metadata
- Resource Attributes
- Provider Types
- Provision Request Supported Attributes
- Advanced Provisioning Attributes
- SOAP to REST Mapping guide
- Deprecated Types
- Deprecated Resource Attributes

- Authentication
- HTTP Headers
- Listing and Querying Collections and Sub-Collections
- Collection Queries:
- Sub-Collection Queries

This document presents the currently available features and capabilities of the REST API.

Authentication

Type	Mechanism
Basic Authentication	Basic HTTP Authorization with user and password
Token Based Authentication	
- Acquiring Token	/api/auth with Basic Authentication
- Authenticating with Token	X-Auth-Token Header

HTTP Headers

Header	Value
Authorization	Basic base64_encoded(user:password)
X-Auth-Token	Token provided by /api/auth
X-MIQ-Group	Group to authorize externally authenticated user
Accept	application/json
Content-Type	application/json

Listing and Querying Collections and Sub-Collections

Feature	Path
Listing Available Collections	/api
Listing Collections	/api/<collection>
Listing Sub-Collections	/api/<collection>/<id>/<sub-collection>

Querying Capability	Query Parameters	
Paging	offset, limit	
Sorting	sort_by=attr, sort_order=asc	desc
	Can sort by database columns and SQL friendly virtual attributes	
Filtering	filter[]=...	
Querying by Tag	i.e. by_tag=/department/finance	
Querying by Multiple Tags	i.e. by_tag=/department/finance,/department/engineering	
Expanding Results	expand=<what>, i.e. expand=resources,tags,service_templates,...	
Selecting Attributes	attributes=<attr1>,<attr2>,... i.e. attributes=id,name,type,...	

Querying Capability	Query Parameters	
	Attributes can be:	
	Database columns	
	Virtual attributes	
	Relationships	

Example Queries

Collection Queries:

For a complete list of supported primary collections please refer to the Primary Collections

Sub-Collection Queries

When applicable, the following subcollections are available:

Sub-Collection	URL
Tagging	/api/<collection>/:id/tags
Policies	/api/<collection>/:id/policies
Policy Profiles	/api/<collection>/:id/policy_profiles
Service Requests	/api/service_templates/:id/service_requests
Request Tasks	
	/api/service_requests/:id/request_tasks
	/api/automation_requests/:id/request_tasks
	/api/provision_requests/:id/request_tasks
Request Tasks can also be accessed via the tasks alias	
	/api/service_requests/:id/tasks
	/api/automation_requests/:id/tasks
	/api/provision_requests/:id/tasks

- Collection Metadata
  - Optional Metadata

Collection Metadata

Collection specific metadata can be queried by sending an HTTP OPTIONS request on the specific collection:

```
OPTIONS /api/:collection
```

What gets returned is the following structure:

```
{
  "attributes": [
  ],
  "virtual_attributes": [
  ],
  "relationships": [
  ],
  "subcollections": [
  ],
  "data": {
  }
}
```

Entry	Description
attributes	Physical attributes of the collection
virtual_attributes	Virtual attributes of the collection
relationships	Physical or Virtual associations to other resources
subcollections	Subcollections that can be queried
data	Optional meta-data for the collection

The following is an example for the **groups** collection:

```
OPTIONS /api/groups
```

```
{
  "attributes": [
```

```
    "created_on",
    "description",
    "group_type",
    "id",
    "sequence",
    "settings",
    "tenant_id",
    "updated_on"
  ],
  "virtual_attributes": [
    "allocated_memory",
    "allocated_storage",
    "allocated_vcpu",
    "custom_1",
    "custom_2",
    "custom_3",
    "custom_4",
    "custom_5",
    "custom_6",
    "custom_7",
    "custom_8",
    "custom_9",
    "href_slug",
    "miq_user_role_name",
    "provisioned_storage",
    "read_only",
    "region_description",
    "region_number",
    "user_count"
  ],
  "relationships": [
    "active_vms",
    "custom_action_buttons",
    "custom_actions",
    "custom_attributes",
    "custom_button_sets",
    "custom_buttons",
    "entitlement",
    "miq_custom_attributes",
    "miq_report_results",
    "miq_reports",
    "miq_templates",
    "miq_user_role",
    "miq_widget_contents",
    "miq_widget_sets",
    "taggings",
    "tags",
    "tenant",
    "users",
    "vms"
  ],
  "subcollections": [
    "tags"
  ],
  "data": {
  }
}
```

Optional Metadata

Most collections do not declare additional metadata as shown by the empty *data* hash entry above. Optional metadata is made available as needed for API clients. They are generally used to provide additional information needed for user-interfaces, help complete or validate data for additional API calls, etc.

Currently, optional metadata is made available for the following collections:

Collection
/api/authentications
/api/clusters
/api/container_deployments
/api/custom_buttons
/api/generic_object_definitions
/api/hosts
/api/providers

There is no enforced structure for the *data* element as each collection may need to provide different types of metadata. The following is a partial meta-data example:

```
OPTIONS /api/generic_object_definitions
```


```
{
  "attributes": [
    "created_at",
    "description",
    "id",
    "name",
    "properties",
    "updated_at"
  ],
  "virtual_attributes": [
    "generic_objects_count",
    "href_slug",
    "region_description",
    "region_number"
  ],
  "relationships": [
    "custom_action_buttons",
    "custom_actions",
    "custom_button_sets",
    "custom_buttons",
    "generic_objects",
    "picture"
  ],
  "subcollections": [
    "generic_objects"
  ],
  "data": {
    "allowed_association_types": [
      [
        "Availability Zone",
        "AvailabilityZone"
      ],
      ...
      [
        "VM and Instance",
        "Vm"
      ],
      [
        "Zone",
        "Zone"
      ]
    ],
    "allowed_types": [
      [
        "Boolean",
        "boolean"
      ],
      [
        "Date/Time",
        "datetime"
      ],
      [
        "Float",
        "float"
      ],
      [
        "Integer",
        "integer"
      ],
      [
        "String",
        "string"
      ],
      [
        "Time",
        "time"
      ]
    ]
  }
}
```


- Resource Attributes
  - Actions
    - Required attributes:
    - Optional attributes:
  - Alert Definitions
    - Required attributes:
    - Optional attributes:

- Authentications
  - Required attributes:
  - Optional attributes:
- Category Attributes
  - Required attributes:
  - Optional attributes:
- Chargeback Rate Attributes
  - Required attributes:
  - Optional attributes:
- Conditions
  - Required attributes:
  - Optional attributes:
- Configuration Script Sources
  - Required attributes:
  - Optional attributes:
- Generic Objects
  - Required attributes:
  - Required attributes:
  - Optional attributes:
- Generic Object Definitions
  - Required attributes:
  - Optional attributes:
- Group Attributes
  - Required attributes:
  - Optional attributes:
    - Filters hash:
- Orchestration Templates
  - Required attributes:
  - Optional attributes:
- Policies
  - Required attributes:
- Provider Attributes
  - Required attributes:
  - Optional attributes:
- Role Attributes
  - Required Attributes:
  - Optional Attributes:
- Service Attributes
  - Required attributes:
  - Optional attributes:
- Service Catalog Attributes
  - Optional attributes:
- Service Dialogs
  - Required attributes:
  - Optional attributes:
- Tag Attributes
  - Required attributes:
- Tenant Attributes
  - Required attributes:
  - Optional attributes:
- User Attributes
  - Required attributes:
  - Optional attributes:

Resource Attributes

This page lists the available attributes when creating resources with the REST API

- Actions
- Alert Definitions
- Authentications

- Categories
- Chargeback Rates
- Conditions
- Configuration Script Sources
- Generic Objects
- Generic Object Definitions
- Groups
- Orchestration Templates
- Policies
- Providers
- Roles
- Services
- Service Catalogs
- Service Dialogs
- Tags
- Tenants
- Users

Actions

`POST /api/actions`

Required attributes:

Attribute	Type	Description
action_type	string	Action Type
description	string	Description (unique)

Optional attributes:

Attribute	Type	Description
name	string	Name
options	hash	Options

Alert Definitions

`POST /api/alert_definitions`

Required attributes:

Attribute	Type	Description
description	string	Description
db	string	Database
expression	hash	Expression
options	hash	Options

Optional attributes:

Attribute	Type	Description
enabled	boolean	Enable the Alert Definition or not - Defaults to true

Authentications

`POST /api/authentications`

Required attributes:

Attribute	Type	Description
manager_resource	hash	**href** of the provider

Optional attributes:

Attribute	Type	Description
type	string	Authentication class

Note:

For a complete list of authentication classes supported and the related attributes for each type, send the *OPTIONS /api/authentications* request.

Category Attributes

`POST /api/categories`

Required attributes:

Attribute	Type	Description
name	string	Name
description	string	Description

Optional attributes:

Attribute	Type	Description
example_text	string	Detailed description
show	boolean	Show Category flag
single_value	boolean	Single Value flag
syntax	string	Syntax of value, **string**, **integer** or **boolean**

Chargeback Rate Attributes

`POST /api/rates`

Required attributes:

Attribute	Type	Description
chargeback_rate_id	integer	Reference to parent Chargeback. `<chargeback_rate>` can be used instead.
group	string	Group rate belongs to, i.e. **cpu**, **memory**, **net_io**, **disk_io**, etc.
source	string	The input value for calculation, i.e. **allocated**, **used**, etc.
chargeback_tiers	Array of hashes	Each tier is represented by hash `{“start”: 0.0, “finish”: ‘infinity’, “fixed_rate”: 2.0, “variable_rate”: 0.0}]

Optional attributes:

Attribute	Type	Description
description	string	Description of the chargeback rate
enabled	boolean	Rate enabled flag
friendly_rate	string	Friendly name of the rate
metric	string	Metrics, i.e. **derived_memory_available**, etc.
per_time	string	Measured per time, **hourly**, **daily**, **weekly**, **monthly** or **yearly**
per_unit	string	Measured per unit, i.e. **megabytes**, **gigabytes**, etc.
chargeback_rate	hash	`id` or `href` from `api/chargebacks` collection.
detail_currency	hash	`id` or `href` from `api/currencies` collection. Details of related currency.
chargeable_field	hash	`id` or `href` from `api/chargeable_fields` collection. Record representing metric. Contains equivalent columns: `metric, group, source`

Conditions

`POST /api/conditions`

Required attributes:

Attribute	Type	Description

Attribute	Type	Description
description	string	Condition Description
expression	hash	Expression
modifier	string	Modifier
towhat	string	Class name

Optional attributes:

Attribute	Type	Description
name	string	Condition name

Configuration Script Sources

`POST /api/configuration_script_sources`

Required attributes:

Attribute	Type	Description
manager_resource	hash	**href** or **id** of the provider

Optional attributes:

Attribute	Type	Description
description	string	Description
name	string	Project Name
related	hash	Related attributes and values

Generic Objects

`POST /api/generic_objects`

Required attributes:

Attribute	Type	Description
name	string	Name of the generic object
generic_object_definitio n	hash	Generic object definition to create new generic object for, requires an **href** or **id** attribute in the hash

`POST /api/generic_object_definitions/:id/generic_objects`

Required attributes:

Attribute	Type	Description
name	string	Name of the generic object

Optional attributes:

Attribute	Type	Description
uid	string	Uid string
property_attributes	hash	Attribute/Value pairs to define for the generic object
associations	hash	Hash containing association keys each being an array of resource references

Generic Object Definitions

Required attributes:

Attribute	Type	Description
name	string	Name of the generic object definition

Optional attributes:

Attribute	Type	Description
description	string	Description for the generic object definition
properties	hash	Hash containing the **attributes**, **associations** and **methods** of the generic object definition

Attribute	Type	Description
picture	hash	Picture to use, either an **href** or **id** of the picture resource, or the actual picture including the **extension** and Base64 encoded **content**

Group Attributes

`POST /api/groups`

Required attributes:

Attribute	Type	Description
description	string	Description
role	hash	Role the group is assigned to, requires an **href**, **id** or **name** attribute in the hash
tenant	hash	Tenant the group belongs to, requires an **href** or **id** attribute in the hash

Optional attributes:

Attribute	Type	Description
filters	hash	Hash that contains the **belongsto** and **managed** arrays of filters to assign to the group
filter_expression	hash	Hash representing the expression used for filtering by Tag

Filters hash:

The `filters` hash can include the following keys:

- **belongsto**
  - Array of tags representing the following types of objects:
    - VMs & Templates
    - Clusters, Datastores, Hosts, Managers & Providers
- **managed**
  - Array of tag arrays representing:
    - My Company Tags

Example:

```
"filters" : {
  "managed" : [
    ["/managed/location/ny"],
    ["/managed/environment/prod"]
  ],
  "belongsto" : [
    "/belongsto/ExtManagementSystem|dev-vc60/EmsFolder|Datacenters/EmsFolder|dev-vc60-
DC/EmsFolder|vm/EmsFolder|Alberto-Dev,
    "/belongsto/ExtManagementSystem|dev-vc60/EmsFolder|Datacenters/EmsFolder|dev-vc60-
DC/EmsFolder|host/EmsCluster|dev-vc60-cluster"
  ]
}
```

Alternatively, company tags can be represented with an expression via the *filter_expression* instead of the *managed* tag array in the *filters* element as follows:

Example with Tags based on expression:

```
"filters" : {
  "managed" : null,
  "belongsto" : [
    "/belongsto/ExtManagementSystem|dev-vc60/EmsFolder|Datacenters/EmsFolder|dev-vc60-
DC/EmsFolder|vm/EmsFolder|Alberto-Dev,
    "/belongsto/ExtManagementSystem|dev-vc60/EmsFolder|Datacenters/EmsFolder|dev-vc60-
DC/EmsFolder|host/EmsCluster|dev-vc60-cluster"
  ]
},
"filter_expression" : {
  "exp": {
    "and": [
      {
        "CONTAINS": {
          "tag": "managed-location",
          "value": "ny"
```

```
        }
      },
      {
        "CONTAINS": {
          "tag": "managed-environment",
          "value": "prod"
        }
      }
    ]
  }
}
```

Orchestration Templates

`POST /api/orchestration_templates`

Required attributes:

Attribute	Type	Description
name	string	Name
content	text	Template content

Optional attributes:

Attribute	Type	Description
type	string	Type
description	string	Description
draft	boolean	**true** or **false**
ems_id	integer	Id of the provider
orderable	boolean	**true** or **false**

Policies

`POST /api/policies`

Required attributes:

Attribute	Type	Description
name	string	Name
description	string	Description
mode	string	Mode (i.e. *compliance*)
towhat	string	Target Class
condition_ids	array	Ids of Conditions to assign to the Policy
policy_contents	array	Array of hashes including *event_id* and *actions*, an array of hashes comprising of an *action_id* and *opts*.

Provider Attributes

`POST /api/providers`

Required attributes:

Attribute	Type	Description
name	string	Name of the provider
type	string	Provider class
hostname	string	Hostname of the Provider
credentials or connection_configurations	hash or array	Credentials information for the provider either via the **credentials** hash or **connection_configurations** array

Optional attributes:

Attribute	Type	Description
api_version	string	API Version for communicating with the Provider
uid_ems	string	Domain for OpenStack provider. Required when “api_version” is “v3”
certificate_authority	string	CA for the Provider
connection_configurations	array	Endpoints for the Provider
credentials	hash	Credentials to use for communicating with the Provider, see Provider Support for examples
host_default_vnc_port_start	integer	Starting VNC port

Attribute	Type	Description
host_default_vnc_port_end	integer	Ending VNC port
ipaddress	string	IP Address of the Provider
port	string	Port of the Provider
realm	string	Realm of the Provider
security_protocol	string	Security Protocol
url	string	URL of the Provider
zone	hash	In which zone to add provider to, hash needs to include an **href** or **id** attribute.

Credentials hash specification attributes are:

Type	Auth Types	Auth Attributes
ManageIQ::Providers::Amazon::CloudManager	default	userid, password
ManageIQ::Providers::Atomic::ContainerManager	default, password, bearer	userid, password
ManageIQ::Providers::Azure::CloudManager	default	userid, password
ManageIQ::Providers::Kubernetes::ContainerManager	default, password, bearer	userid, password
ManageIQ::Providers::Microsoft::InfraManager	default	userid, password
ManageIQ::Providers::Openshift::ContainerManager	default, password, bearer	userid, password, auth_key
ManageIQ::Providers::Openstack::InfraManager	default, amqp, ssh_keypair	userid, password, auth_key
ManageIQ::Providers::Openstack::CloudManager	default, amqp	userid, password
ManageIQ::Providers::Redhat::InfraManager	default, metrics	userid, password
ManageIQ::Providers::Vmware::InfraManager	default	userid, password
ManageIQ::Providers::Foreman::Provider	default	userid, password
ManageIQ::Providers::Openstack::Provider	default	userid, password

Role Attributes

`POST /api/roles`

Required Attributes:

Attribute	Type	Description
name	string	Name of user role

Optional Attributes:

Attribute	Type	Description
features	array	Features to assign to the role. Array of { “identifier” OR “href” OR “id” : ... }
settings	hash	Settings for the role, normally: { “restrictions” : { “vms” : “user” OR “user_or_group” } }

Service Attributes

`POST /api/services`

Required attributes:

Attribute	Type	Description
name	string	Name

Optional attributes:

Attribute	Type	Description
description	string	Description
orchestration_template	hash	**href** or **id** of the Orchestration Template
orchestration_manager	hash	**href** or **id** of the Orchestration Provider
parent_service	hash	Parent service **href** or **id** reference
type	string	Type of Service

Service Catalog Attributes

`POST /api/service_catalogs`

Attribute	Type	Description
name	string	Name of service catalog
description	string	Description of service catalog

Optional attributes:

Attribute	Type	Description
service_templates	array	Array of Service Template hashes to assign to the new Service Catalog. Hash entries to include the **href** of the service_templates to assign and must not currently be assigned to any Service Catalog.

Service Dialogs

`POST /api/service_dialogs`

Required attributes:

Attribute	Type	Description
label	string	Label
dialog_tabs	array	Dialog content - array of dialog tabs

Optional attributes:

Attribute	Type	Description
description	string	Description

Tag Attributes

`POST /api/tags`

Required attributes:

Attribute	Type	Description
name	string	Name of tag
description	string	Description of tag
category	hash	Category identifier for which the tag is getting added to, requires an **href**, **id** or **name** attribute in the hash

Note:

When adding with a **create** action on the tags subcollection of the category via /api/categories/:id/tags, the category should not be specified in the resource.

Tenant Attributes

`POST /api/tenants`

Required attributes:

Attribute	Type	Description
name	string	Name of Tenant
parent	hash	Parent tenant identifier for which the sub-tenant is getting added to, requires an **href** or **id** attribute in the hash

Optional attributes:

Attribute	Type	Description
description	string	Description of Tenant
domain	string	Domain
logo_file_name	string	Logo file name, must be available via /uploads/**logo_file_name**
login_logo_file_name	string	Login Logo file name, must be available via /uploads/**login_logo_file_name**
login_text	string	Login text
subdomain	string	Subdomain

User Attributes

`POST /api/users`

Required attributes:

Attribute	Type	Description
userid	string	Username
password	string	Password
name	string	Full Name

Attribute	Type	Description
group	hash	Group the user belongs to, requires a **description**, **href** or **id** attribute in the hash

Optional attributes:

Attribute	Type	Description
email	string	E-mail Address

- Provider Types

Provider Types

The provider types available with earlier appliances have changed with this release. This is what gets specified with the *type* attribute when creating a new provider:

```
POST /api/providers

{
  "name" : "vCenter 5",
  "type" : "ManageIQ::Providers::Vmware::InfraManager",
  "hostname" : "my_vcenter_50",
  ...
}
```

The following table lists the type from previous appliance releases (*v2.0.0* of the REST API), and their mapping to the new *v2.1.0* types:

v2.0.0 Provider Type	v2.1.0 Provider Type
EmsAmazon	ManageIQ::Providers::Amazon::CloudManager
	ManageIQ::Providers::Atomic::ContainerManager (**new**)
	ManageIQ::Providers::Azure::CloudManager (**new**)
EmsKubernetes	ManageIQ::Providers::Kubernetes::ContainerManager
EmsMicrosoft	ManageIQ::Providers::Microsoft::InfraManager
	ManageIQ::Providers::Openshift::ContainerManager (**new**)
EmsOpenstackInfra	ManageIQ::Providers::Openstack::InfraManager
EmsOpenstack	ManageIQ::Providers::Openstack::CloudManager
EmsRedhat	ManageIQ::Providers::Redhat::InfraManager
EmsVmware	ManageIQ::Providers::Vmware::InfraManager

And when *provider-class* is set to **provider**, the following mapping table applies:

v2.0.0 Provider Type	v2.1.0 Provider Type
ProviderForeman	ManageIQ::Providers::Foreman::Provider
ProviderOpenstack	ManageIQ::Providers::Openstack::Provider

- Provision Request Supported Attributes
  - Requester Attributes
  - Custom Attributes
  - Service Catalog Attributes
  - Schedule Attributes
  - Network Attributes
  - Hardware Attributes
  - Environment Attributes

Provision Request Supported Attributes

Attribute Class	Provision Request Attribute Group
Requester Attributes	requester
Custom Attributes	vm_fields
Service Catalog Attributes	vm_fields
Schedule Attributes	vm_fields
Network Attributes	vm_fields
Hardware Attributes	vm_fields
Environment Attributes	additional_values

Requester Attributes

Attribute	Type	Description
owner_phone	string	Phone
owner_country	string	Country/Region
owner_phone_mobile	string	Mobile phone
owner_title	string	Title
owner_first_name	string	First name
owner_manager	string	Manager name
owner_address	string	Address
owner_company	string	Company
owner_last_name	string	Last name
owner_manager_email	string	Manager e-mail address
owner_city	string	City
owner_department	string	Department
owner_load_ldap	button	Look up LDAP e-mail address
owner_manager_phone	string	Manager phone
owner_state	string	State
owner_office	string	Office
owner_zip	string	Zip code
owner_email	string	E-Mail
request_notes	string	Notes

Custom Attributes

Attribute	Type	Description	Values	Default
dns_servers	string	DNS server list		
sysprep_organization	string	Organization		
sysprep_password	string	New Administrator Password		
sysprep_custom_spec	string	Name		
sysprep_server_license_mode	string	Identification	perServer, perSeat	
ldap_ous	string	LDAP Group		
sysprep_timezone	string	Timezone		
dns_suffixes	string	DNS Suffix list		
sysprep_product_id	string	ProductID		
sysprep_identification	string	Identification	domain, workgroup	
sysprep_per_server_max_connections	string	Maximum Connections		5
sysprep_computer_name	string	Computer Name		
sysprep_workgroup_name	string	Workgroup Name		WORKGROUP
sysprep_spec_override	boolean	Override Specification		false
addr_mode	string	Address Mode	static, dhcp	dhcp
linux_host_name	string	Computer Name		
sysprep_domain_admin	string	Domain Admin		
sysprep_change_sid	boolean	Change SID		true
sysprep_domain_name	string	Domain Name		
sysprep_upload_file	string	Upload		
gateway	string	Gateway		
ip_addr	string	IP Address		
linux_domain_name	string	Domain Name		
sysprep_domain_password	string	Domain Password		
sysprep_auto_logon	boolean	Auto Logon		true
sysprep_enabled	string	Customize		disabled
sysprep_delete_accounts	boolean	Delete Accounts		false
sysprep_upload_text	string	Sysprep Text		
wins_servers	string	WINS Server list		
subnet_mask	string	Subnet Mask		
sysprep_full_name	string	Full Name		
sysprep_auto_logon_count	integer	Auto Logon Count	1, 2, 3	1
customization_template_id	integer	Script Name		
root_password	string	Root Password		
hostname	string	Host Name		

Attribute	Type	Description	Values	Default
customization_template_script	string	Script Text		

Service Catalog Attributes

Attribute	Type	Description	Default
number_of_vms	integer	Count. *Max*: 50	1
vm_description	string	VM Description. *Max length*: 100	
vm_prefix	string	VM Name Prefix/Suffix	
src_vm_id	integer	Name	
vm_name	string	VM Name	
pxe_image_id	string	Image	
pxe_server_id	integer	Server	
host_name	string	Host Name	
provision_type	string	Provision Type	vmware
linked_clone	boolean	Linked Clone	false
snapshot	string	Snapshot	
vm_filter	integer	Filter	

Schedule Attributes

Attribute	Type	Description	Values	Default
schedule_type	string	When to Provision	schedule, immediately (*On Approval*)	immediately
vm_auto_start	boolean	Power on virtual machines after creation		true
schedule_time	time	Time to Provision on		
retirement	integer	Time until Retirement	0 (*Indefinite*), 1.month, 3.months, 6.months	0
retirement_warn	integer	Retirement Warning	1.week, 2.weeks, 30.days	1.week

Network Attributes

Attribute	Type	Description
vlan	string	vLan
mac_address	string	MAC Address

Hardware Attributes

Attribute	Type	Description	Values	Default
disk_format	string	Disk Format	thick, thin, unchanged	unchanged
cpu_limit	integer	CPU (MHz)		
memory_limit	integer	Memory (MB)		
number_of_sockets	integer	Number of Sockets	1, 2, 4, 8	1
cores_per_socket	integer	Cores per Socket	1, 2, 4, 8	1
cpu_reserve	integer	CPU (MHz)		
vm_memory	string	Memory (MB)	1024, 2048, 4096	1024
memory_reserve	integer	Memory (MB)		
network_adapters	integer	Network Adapters	1, 2, 3, 4	1

Environment Attributes

These attributes cannot be passed directly. To use these attributes, provide them in the **additional_values** attribute group and have customization methods use them and modify the request accordingly.

Attribute	Type	Description	Values	Default
new_datastore_grow_increment	integer	Grow Increment (GB)		
new_datastore_create	boolean	Create Datastore		false
placement_cluster_name	integer	Name		
new_datastore_aggregate	string	Aggregate		
new_datastore_max_size	integer	Max Size (GB)		
new_datastore_storage_controller	string	Controller		
cluster_filter	integer	Filter		
host_filter	integer	Filter		
ds_filter	integer	Filter		
new_datastore_volume	string	Volume		

Attribute	Type	Description	Values	Default
placement_host_name	integer	Name		
placement_ds_name	integer	Name		
new_datastore_fs_type	string	FS Type	NFS, VMFS	NFS
rp_filter	integer	Filter		
new_datastore_thin_provision	string	Thin Provision		
placement_auto	boolean	Choose Automatically		false
new_datastore_size	integer	Size (GB)		
new_datastore_autogrow	string	Autogrow		false
placement_folder_name	integer	Name		
new_datastore_name	string	Name		
placement_rp_name	integer	Name		
placement_dc_name	integer	Name		

- Advanced Provisioning Attributes
  - Defining new vLans adapters
  - Defining new SCSI Controller
  - Defining new Disks
  - Cloning Type
  - Limit Template Selection

Advanced Provisioning Attributes

This appendix lists parameters that apply to REST API initiated provisioning via /api/provision_requests.

- Defining new vLans adapters
- Defining new SCSI Controller
- Defining new Disks
- Cloning Type
- Limit Template Selection

Defining new vLans adapters

These attributes allow adding new vLan adapters during the provisioning request.

Attribute	Type	Required	Values	Default
network##	string	Required	Network Name	
network##.mac_address	string	Optional	MAC Address	
network##.deviceType	string	Optional	VirtualE1000, VirtualPCNet32, VirtualVmxnet	VirtualPCNet32
network##.connectable.allowGuestControl	boolean	Optional		true
network##.connectable.startConnected	boolean	Optional		true
network##.connectable.connected	boolean	Optional		true

Example:

```
"network0" : "VM Network",
"network0.mac_address" : "00:50:56:57:58:59"
```

Defining new SCSI Controller

The SCSI Controller attributes are optional. If a controller is required, it will be added with the defined defaults. These settings are only required if you need to override a default value.

Attribute	Type	Required	Values	Default
ctrlscsi##	string	Required	VirtualBusLogicController, VirtualLsiLogicController	VirtualLsiLogicController
ctrlscsi##.sharedBus	string	Optional	noSharing, physicalSharing, virtualSharing	noSharing

Note:

The ## value defines the controller bus number

Example:

```
"ctrlscsi00" : "VirtualBusLogicController"
"ctrlscsi00.sharedBus" : "virtualSharing"
```

Defining new Disks

The Disk attributes allow adding disks as part of the provisioning request.

Attribute	Type	Required	Values	Default
diskscsi##	string	Required	“controller_#:controller_position_#:size_in_mb”	
diskscsi##.datastore	string	Optional	Datastore Name	
diskscsi##.filename	string	Optional	Filename	
diskscsi##.connectable.allowGuestControl	boolean	Optional		false
diskscsi##.connectable.startConnected	boolean	Optional		true
diskscsi##.connectable.connected	boolean	Optional		true
diskscsi##.backing.diskmode	string	Optional	persistent, independent_persistent, independent_nonpersistent, nonpersistent	persistent
diskscsi##.backing.split	boolean	Optional		false
diskscsi##.backing.thinProvisioned	boolean	Optional		false
diskscsi##.backing.writeThrough	boolean	Optional		false

Example:

```
"diskscsi0" : "0:0:10"
"diskscsi1" : "0:0:8"
"diskscsi1.datastore" : "storage1"
"diskscsi1.backing.diskmode" : "independent_persistent"
```

Cloning Type

This attribute sets the type of cloning request, either clone to vm or clone to template. It is an attribute to set with the *template_fields* hash.

Attribute	Type	Required	Values	Default
request_type	string	Optional	clone_to_vm, clone_to_template	clone_to_vm

Example:

```
"template_fields" : {
  "guid" : "afe6e8a0-89fd-11e3-b6ac-b8e85646e742",
  "request_type" : "clone_to_template"
}
```

Limit Template Selection

This attribute value limits the template selection to the Datacenters contained in the specified cluster. It is an attribute to be specified with the *additional_values* hash.

Attribute	Type	Required
cluster	string	Optional

Example:

```
"additional_values" : {
  "request_id" : "87458",
  "cluster" : "sales_prod"
}
```


- SOAP to REST Mapping guide
  - Insight Web Services
  - Control Web Services
  - Automate Web Services

SOAP to REST Mapping guide

- Insight Web Services
- Control Web Services
- Automate Web Services

Insight Web Services

SOAP API	HTTP Method	REST API URLs	Actions	Examples
ClusterGetTags	GET	/api/cluster/:id/tags		
DatastoreGetTags	GET	/api/data_stores/:id/tags		
EmsGetTags	GET	/api/providers/:id/tags		
EVMDeleteVmByName	DELETE	/api/vms/:id		Delete VM
	POST	/api/vms/:id	delete	Delete VMs
EVMClusterList	GET	/api/clusters		
EVMDatastoreList	GET	/api/data_stores		
EVMHostList	GET	/api/hosts		
EVMResourcePoolList	GET	/api/resource_pools		
EVMPing	GET	/api		
EVMVmAccounts	GET	/api/vms/:id/accounts		
EVMVmList	GET	/api/vms		
	GET	/api/hosts/:id?attributes=vms		
	GET	/api/hosts?filter[]=guid='....'\&expand=resources\&attributes=vms		
EVMVmSoftware	GET	/api/vms:id/software		
FindClusterById	GET	/api/clusters/:id		
FindClustersById	GET	/api/clusters/:id		
FindDatastoreById	GET	/api/data_stores/:id		
FindDatastoresById	GET	/api/data_stores/:id		
FindEmsByGuid	GET	/api/providers?filter[]=guid='....'		
FindHostByGuid	GET	/api/hosts?filter[]='guid='....'"		
FindHostsByGuid	GET	/api/hosts?filter[]='guid='....'"		
FindResourcePoolById	GET	/api/resource_pools/:id		
FindResourcePoolsById	GET	/api/resource_pools		
	GET	/api/resource_pools?filter[]=id=:id\&filter[]=or id=:id&...		
FindVmByGuid	GET	/api/vms?filter[]=guid='....'		
FindVmsByGuid	GET	/api/vms?filter[]=guid='....'\&filter[]=or guid='....'&...		
GetClusterList	GET	/api/clusters		
	GET	/api/clusters?filter[]=ems_id=:id		
	GET	/api/providers/:id?attributes=ems_clusters		
GetClustersByList	GET	/api/clusters/:id		
	GET	/api/clusters?filter[]=id=:id\&filter[]=or id=:id&...		
GetClustersByTag	GET	/api/clusters?by_tag=:tag_path		
GetDatastoreByList	GET	/api/data_stores		
GetDatastoreByTag	GET	/api/data_stores?by_tag=:tag_path		
GetEmsByList	GET	/api/providers		
	GET	/api/providers/:id		
	GET	/api/providers?filter[]=....\&expand=resources		
GetEmsList	GET	/api/providers		
GetHostsByList	GET	/api/hosts		
	GET	/api/hosts?filter[]=....\&expand=resources		
GetHostList	GET	/api/hosts		
	GET	/api/providers/:id?attributes=hosts		
	GET	/api/providers?filter[]=guid='... .'\&expand=resources\&attributes=hosts		
GetResourcePoolList	GET	/api/providers/:id?attributes=resource_pools		
	GET	/api/providers?filter[]=guid='... .'\&expand=resources\&attributes=resource_pools		
GetResourcePoolsByList	GET	/api/resource_pools		
	GET	/api/resource_pools/:id		
	GET	/api/resource_pools?filter[]=....\&expand=resources		

SOAP API	HTTP Method	REST API URLs	Actions	Examples
GetResourcePoolsByTag	GET	/api/resource_pools?by_tag=:tag_path		
GetTemplatesByTag	GET	/api/templates?by_path=:tag_path		
GetVmList	GET	/api/hosts/:id?attributes=vms		
	GET	/api/hosts?filter[]=guid='....'\&expand=resources\&attributes=vms		
GetVmsByList	GET	/api/vms		
	GET	/api/vms/:id		
	GET	/api/vms?filter[]=....\&expand=resources		
GetVmsByTag	GET	/api/vms?by_path=:tag_path		
HostGetTags	GET	/api/hosts/:id/tags		
	GET	/api/hosts/:id?expand=tags		
	GET	/api/hosts?filter[]=guid='....'\&expand=resources,tags		
ResourcePoolGetTags	GET	/api/resource_pools/:id/tags		
	GET	/api/resource_pools/:id?expand=tags		
VmGetTags	GET	/api/vms/:id/tags		
	GET	/api/vms/:id?expand=tags		

Control Web Services

SOAP API	HTTP Method	REST API URLs	Actions	Examples
ClusterSetTag	POST	/api/clusters/:id/tags	assign	Assigning Tags
DatastoreSetTag	POST	/api/data_stores/:id/tags	assign	Assigning Tags
EmsSetTag	POST	/api/providers/:id/tags	assign	Assigning Tags
EVMActionList	GET	/api/policies/:id/policy_actions		Policy Actions
	GET	/api/policies/:id?expand=policy_actions		
	GET	/api/policies?filter[]=guid='... .'\&expand=resources,policy_actions		
EVMAssignPolicy	POST	/api/hosts/:id/policies	assign	Assigning Policies
EVMConditionList	GET	/api/policies/:id/conditions		Policy Conditions
	GET	/api/policies/:id?expand=conditions		
	GET	/api/policies?filter[]=guid='... .'\&expand=resources,conditions		
EVMEventList	GET	/api/policies/:id/events		Policy Events
	GET	/api/policies/:id?expand=events		
	GET	/api/policies?filter[]=guid='... .'\&expand=resources,events		
EVMGetPolicy	GET	/api/policies		Querying Policies
	GET	/api/policies/:id		
	GET	/api/policies?filter[]=name='....'		
EVMPolicyList	GET	/api/hosts/:id/policies		Policy Management
	GET	/api/hosts/:id?expand=policies		
	GET	/api/hosts?filter[]=guid='... .'\&expand=resources,policies		
EVMVmEventByProperty	POST	/api/vms/:id	add_event	Add VM Event
EVMVmScanByProperty	POST	/api/vms/:id	scan	Scan VM
EVMSmartStart	POST	/api/vms/:id	start	Start VM
EVMSmartStop	POST	/api/vms/:id	stop	Stop VM
EVMSmartSuspend	POST	/api/vms/:id	suspend	Suspend VM
EVMUnassignPolicy	POST	/api/hosts/:id/policies	unassign	Unassigning Policies
EVMVmRsop	POST	/api/vms/:id/policies/:id	resolve	Resolving Policies
	POST	/api/vms/:id/policy_profiles/:id	resolve	Resolving Policy Profiles
HostSetTag	POST	/api/hosts/:id/tags	assign	Assigning Tags
ResourcePoolSetTag	POST	/api/resource_pools/:id/tags	assign	Assigning Tags
VmAddCustomAttribute	POST	/api/vms/:id/custom_attributes	add	Adding Custom Attributes
VmAddCustomAttributes	POST	/api/vms/:id/custom_attributes	add	

SOAP API	HTTP Method	REST API URLs	Actions	Examples
VmAddCustomAttributeByFields	POST	/api/vms/:id/custom_attributes	add	
VmDeleteCustomAttribute	POST	/api/vms/:id/custom_attributes	delete	Deleting Custom Attributes
VmDeleteCustomAttributes	POST	/api/vms/:id/custom_attributes	delete	
VmSetTag	POST	/api/vms/:id/tags	assign	Assigning Tags
VmSetOwner	POST	/api/vms/:id	set_owner	Set Owner of VM

Automate Web Services

SOAP API	HTTP Method	REST API URLs	Actions	Examples
CreateAutomationRequest	POST	/api/automation_requests	create	Create Automation Request, Create Automation Requests
EVMProvisionRequestEx	POST	/api/provision_requests	create	Create Provision Request, Create Provision Requests
GetAutomationRequest	GET	/api/automation_requests/:id		
GetAutomationTask	GET	/api/automation_requests/:id?expand=request_tasks		
	GET	/api/automation_requests/:id/request_tasks/:id		
GetVmProvisionRequest	GET	/api/provision_requests/:id		
	GET	/api/provision_requests/:id?expand=request_tasks		
VmProvisionRequest	POST	/api/provision_requests	create	Create Provision Request, Create Provision Requests

- Deprecated Types
  - API v2.4.0 ⇒ v3.0.0

Deprecated Types

This page lists the types that have been deprecated in previous releases and have been updated to newer types. They are *temporarily* supported in the listed releases but will be removed in future releases.

API v2.4.0 ⇒ v3.0.0

For Orchestration Templates:

Old Type	New Type
OrchestrationTemplateAzure	ManageIQ::Providers::Azure::CloudManager::OrchestrationTemplate
OrchestrationTemplateHot	ManageIQ::Providers::Openstack::CloudManager::OrchestrationTemplate
OrchestrationTemplateCfn	ManageIQ::Providers::Amazon::CloudManager::OrchestrationTemplate
OrchestrationTemplateVnfd	ManageIQ::Providers::Openstack::CloudManager::VnfdTemplate

- Deprecated Resource Attributes
  - API v2.2.0 ⇒ v2.3.0

Deprecated Resource Attributes

This page lists the resource attributes that have been deprecated in earlier releases and have been either renamed to newer attribute names or simply removed.

API v2.2.0 ⇒ v2.3.0

For Hardware:

Old Attribute Name	New Attribute Name
numvcpus	cpu_sockets
memory_cpu	memory_mb

For Metrics:

Old Attribute Name	New Attribute Name
v_derived_logical_cpus_used	v_derived_cpu_total_cores_used

For Provision Workflows:

Old Attribute Name	New Attribute Name
allowed_datastore_aggregate	**removed**
allowed_datastore_storage_controller	**removed**

For all:

Old Attribute Name	New Attribute Name
cores_per_socket	cpu_cores_per_socket
logical_cpus	cpu_total_cores
aggregate_logical_cpus	aggregate_cpu_total_cores

Infrastructure management API Examples

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Foundation
- Automation Requests
- Providers
- Provisioning Requests
- Service Catalogs
- Service Templates
- Services
- VM Management
- Tag Management
- Usage Flows

Infrastructure management API Examples - Foundation

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Querying Examples
- Paging Queries

- Example Queries:
  - Query All VMs
  - Query A Specific VM
  - Query all VMs, but return only return name and vendor
  - Query VMs named sample* and return name and vendor
  - Query VMs but only return the first 500
  - Query VMs but return the second 500
  - Query first 1000 VMs, named test*, get name, vendor and guid and sort by name in ascending order
  - Query Services tagged for the finance department
  - Query Services tagged with the finance or engineering department
  - Get details on the Tags of the first service
  - Get details of the first service catalog and related details on the assigned service templates
  - Get services requests created from a specific template and get details on all related request tasks.
  - Get a specific provision request with expanded details on the associated provision request tasks

Example Queries:

Query All VMs

```
GET /api/vms
```

Query A Specific VM

```
GET /api/vms/1386
```

Query all VMs, but return only return name and vendor

```
GET /api/vms?expand=resources&attributes=name,vendor
```

Query VMs named sample* and return name and vendor

```
GET /api/vms?expand=resources&attributes=name,vendor&filter[]=name='sample%'
```

Query VMs but only return the first 500

```
GET /api/vms?offset=0&limit=500
```

Query VMs but return the second 500

```
GET /api/vms?offset=500&limit=500
```

Query first 1000 VMs, named test*, get name, vendor and guid and sort by name in ascending order

```
GET /api/vms?offset=0&limit=1000&filter[]=name='test%'
    &expand=resources&attributes=name,vendor,guid
    &sort_by=name&sort_order=asc
```

Query Services tagged for the finance department

```
GET /api/services?by_tag=/department/finance
```

Query Services tagged with the finance or engineering department

```
GET /api/services?by_tag=/department/finance,/department/engineering
```

Get details on the Tags of the first service

```
GET /api/services/1/tags?expand=resources
```

Get details of the first service catalog and related details on the assigned service templates

```
GET /api/service_catalogs/1?expand=service_templates
```

Get services requests created from a specific template and get details on all related request tasks.

```
GET /api/service_templates/25/service_requests?expand=resources,request_tasks
```

Get a specific provision request with expanded details on the associated provision request tasks

```
GET /api/provision_requests/120?expand=request_tasks
```


- Paging Queries
  - Request:
  - Response:
  - Request:
  - Response:
  - Request:
  - Response:
  - Request:
  - Response:

Paging Queries

In this series of requests, we will show the paging queries and the expected responses for each subsequent page.

Request:

```
GET /api/vms?offset=0&limit=500
    &sort_order=asc&sort_by=name
    &expand=resources&attributes=name
```

Response:

```
{
  "name": "vms",
  "count": 1912,
  "subcount": 500,
  "resources": [
    {
```

```
      "href": "http://localhost:3000/api/vms/176",
      "id": "176",
      "name": "53 Zone1"
    },
    ...
    {
      "href": "http://localhost:3000/api/vms/1575",
      "id": "1575",
      "name": "VmEmpty-3de98f0f-c6f3-4f8b-a932-554713a61067"
    }
  ],
  "actions": [
  ]
}
```

Request:

```
GET /api/vms?offset=500&limit=500
    &sort_order=asc&sort_by=name
    &expand=resources&attributes=name
```

Response:

```
{
  "name": "vms",
  "count": 1912,
  "subcount": 500,
  "resources": [
    {
      "href": "http://localhost:3000/api/vms/1574",
      "id": "1574",
      "name": "VmEmpty-3e13ff43-6907-4a22-8f95-58aeb1bffa0b"
    },
    ...
    {
      "href": "http://localhost:3000/api/vms/1076",
      "id": "1076",
      "name": "VmEmpty-9a885181-7771-4f91-9805-245c7606d833"
    }
  ],
  "actions": [
  ]
}
```

Request:

```
GET /api/vms?offset=1000&limit=500
    &sort_order=asc&sort_by=name
    &expand=resources&attributes=name
```

Response:

```
{
  "name": "vms",
  "count": 1912,
  "subcount": 500,
  "resources": [
    {
      "href": "http://localhost:3000/api/vms/1074",
      "id": "1074",
      "name": "VmEmpty-9ab9e101-92b0-4b6b-864e-e196538da8a8"
    },
    ...
    {
      "href": "http://localhost:3000/api/vms/575",
      "id": "575",
      "name": "VmEmpty-f251f135-01c8-4d44-b8e1-37b30844a9dd"
    }
  ],
  "actions": [
  ]
}
```

Request:

```
GET /api/vms?offset=1500&limit=500
    &sort_order=asc&sort_by=name
    &expand=resources&attributes=name
```

Response:

```
{
  "name": "vms",
  "count": 1912,
  "subcount": 412,
  "resources": [
    {
      "href": "http://localhost:3000/api/vms/574",
      "id": "574",
      "name": "VmEmpty-f28912f3-b096-487f-9763-97b39b67364b"
    },
    ...
    {
      "href": "http://localhost:3000/api/vms/1907",
      "id": "1907",
      "name": "yy_vm"
    }
  ],
  "actions": [
  ]
}
```

Note:

In this last request, the **subcount** was less than the requested page size thus denoting the last page of data being returned.

Infrastructure management API Examples - Automation Requests

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Create an Automation Request
- Create multiple Automation Requests
- Trigger a single Automation Request
  - Request:
  - Request:
  - Response:

Trigger a single Automation Request

In the automation requests:

- version defaults to “1.1” if not specified.
- user_name defaults to the REST API authenticated user if not specified.

Request:

```
POST /api/automation_requests
```

```
{
  "version" : "1.1",
  "uri_parts" : {
    "namespace" : "System",
    "class" : "Request",
    "instance" : "InspectME",
    "message" : "create"
  },
  "parameters" : {
    "var1" : "xxxxx",
    "var2" : "yyyyy",
    "var3" : 1024,
    "var4" : true,
    "var5" : "last value"
  },
  "requester" : {
    "user_name" : "admin",
    "auto_approve" : true
  }
}
```

Optionally, the action based request format is also supported:

Request:

```
POST /api/automation_requests

{
  "action" : "create",
  "resource" : {
    "version" : "1.1",
    "uri_parts" : {
      "namespace" : "System",
      "class" : "Request",
      "instance" : "InspectME",
      "message" : "create"
    },
    "parameters" : {
      "var1" : "xxxxx",
      "var2" : "yyyyy",
      "var3" : 1024,
      "var4" : true,
      "var5" : "last value"
    },
    "requester" : {
      "user_name" : "admin",
      "auto_approve" : true
    }
  }
}
```

Response:

```
{
  "results": [
    {
      "id": "12",
      "href" : "http://localhost:3000/api/automation_requests/12",
      "description": "Automation Task",
      "approval_state": "approved",
      "type": "AutomationRequest",
      "created_on": "2015-04-16T21:49:55Z",
      "updated_on": "2015-04-16T21:49:55Z",
      "requester_id": "1",
      "requester_name": "Administrator",
      "request_type": "automation",
      "request_state": "pending",
      "status": "Ok",
      "options": {
        "message": "create",
        "namespace": "System",
        "class_name": "Request",
        "instance_name": "InspectME",
        "user_id": "1",
        "attrs": {
          "var1": "xxxxx",
          "var2": "yyyyy",
          "var3": 1024,
          "var4": true,
          "var5": "last value",
          "userid": "admin"
        }
      },
      "userid": "admin"
    }
  ]
}
```


- Trigger multiple Automation Requests
  - Request:
  - Response:

Trigger multiple Automation Requests

In the automation requests:

- version defaults to “1.1” if not specified.
- user_name defaults to the REST API authenticated user if not specified.

Request:

```
POST /api/automation_requests
```

```
{
  "action" : "create",
  "resources" : [
    {
      "version" : "1.1",
      "uri_parts" :  {
        "namespace" : "System",
        "class" : "Request",
        "instance" : "InspectME",
        "message" : "create"
      },
      "parameters" : {
        "vm_name" : "test_1",
        "var2" : "yyyyy",
        "var3" : 1024,
        "var4" : true,
        "var5" : "last value"
      },
      "requester" :  {
        "user_name" : "jdoe",
        "auto_approve" : true
      }
    },
    {
      "uri_parts" :  {
        "namespace" : "System",
        "class" : "Request",
        "instance" : "InspectME",
        "message" : "create"
      },
      "parameters" : {
        "vm_name" : "test_2",
        "vm_memory" : 1024,
        "memory_limit" : 16384
      },
      "requester" :  {
        "auto_approve" : true
      }
    },
    {
      "uri_parts" :  {
        "namespace" : "System",
        "class" : "Request",
        "instance" : "InspectME",
        "message" : "create"
      },
      "parameters" : {
        "vm_name" : "test_3",
        "vm_memory" : 2048,
        "memory_limit" : 16384
      },
      "requester" :  {
        "auto_approve" : true
      }
    },
    {
      "uri_parts" :  {
        "namespace" : "System",
        "class" : "Request",
        "instance" : "InspectME",
        "message" : "create"
      },
      "parameters" : {
        "vm_name" : "test_4",
        "vm_memory" : 4096,
        "memory_limit" : 16384
      },
      "requester" :  {
        "auto_approve" : true
      }
    }
  ]
}
```

Response:

```
{
  "results": [
    {
```

```
      "id": "14",
      "href" : "http://localhost:3000/api/automation_requests/14",
      "description": "Automation Task",
      "approval_state": "approved",
      "type": "AutomationRequest",
      "created_on": "2015-04-16T21:59:42Z",
      "updated_on": "2015-04-16T21:59:42Z",
      "requester_id": "13",
      "requester_name": "aab",
      "request_type": "automation",
      "request_state": "pending",
      "status": "Ok",
      "options": {
        "message": "create",
        "namespace": "System",
        "class_name": "Request",
        "instance_name": "InspectME",
        "user_id": "13",
        "attrs": {
          "vm_name": "test_1",
          "var2": "yyyyy",
          "var3": 1024,
          "var4": true,
          "var5": "last value",
          "userid": "aab"
        }
      },
      "userid": "aab"
    },
    {
      "id": "15",
      "href" : "http://localhost:3000/api/automation_requests/15",
      "description": "Automation Task",
      "approval_state": "approved",
      "type": "AutomationRequest",
      "created_on": "2015-04-16T21:59:42Z",
      "updated_on": "2015-04-16T21:59:42Z",
      "requester_id": "1",
      "requester_name": "Administrator",
      "request_type": "automation",
      "request_state": "pending",
      "status": "Ok",
      "options": {
        "message": "create",
        "namespace": "System",
        "class_name": "Request",
        "instance_name": "InspectME",
        "user_id": "1",
        "attrs": {
          "vm_name": "test_2",
          "vm_memory": 1024,
          "memory_limit": 16384,
          "userid": "admin"
        }
      },
      "userid": "admin"
    },
    {
      "id": "16",
      "href" : "http://localhost:3000/api/automation_requests/16",
      "description": "Automation Task",
      "approval_state": "approved",
      "type": "AutomationRequest",
      "created_on": "2015-04-16T21:59:42Z",
      "updated_on": "2015-04-16T21:59:42Z",
      "requester_id": "1",
      "requester_name": "Administrator",
      "request_type": "automation",
      "request_state": "pending",
      "status": "Ok",
      "options": {
        "message": "create",
        "namespace": "System",
        "class_name": "Request",
        "instance_name": "InspectME",
        "user_id": "1",
        "attrs": {
          "vm_name": "test_3",
          "vm_memory": 2048,
          "memory_limit": 16384,
          "userid": "admin"
```

```
        }
      },
      "userid": "admin"
    },
    {
      "id": "17",
      "href" : "http://localhost:3000/api/automation_requests/17",
      "description": "Automation Task",
      "approval_state": "approved",
      "type": "AutomationRequest",
      "created_on": "2015-04-16T21:59:42Z",
      "updated_on": "2015-04-16T21:59:42Z",
      "requester_id": "1",
      "requester_name": "Administrator",
      "request_type": "automation",
      "request_state": "pending",
      "status": "Ok",
      "options": {
        "message": "create",
        "namespace": "System",
        "class_name": "Request",
        "instance_name": "InspectME",
        "user_id": "1",
        "attrs": {
          "vm_name": "test_4",
          "vm_memory": 4096,
          "memory_limit": 16384,
          "userid": "admin"
        }
      },
      "userid": "admin"
    }
  ]
}
```

Infrastructure management API Examples - Providers

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Create a Provider
- Create a Provider with multiple credentials
- Update a Provider
- Refresh a Provider
- Delete a Provider

- Create Provider
  - Request:
  - Response:

Create Provider

Request:

```
POST /api/providers
```

```
{
  "type"       : "ManageIQ::Providers::Redhat::InfraManager",
  "name"       : "rhevm101",
  "hostname"   : "rhevm101",
  "ipaddress"  : "100.200.300.101",
  "credentials" : {
    "userid"     : "admin_account",
    "password"   : "admin_password"
  }
}
```

Response:

```
{
  "results": [
    {
      "id": "105",
      "href" : "http://localhost:3000/api/providers/105",
      "name": "rhevm101",
```

```
      "hostname": "rhevm101",
      "ipaddress": "100.200.300.101",
      "created_on": "2015-05-05T15:47:41Z",
      "updated_on": "2015-05-05T15:47:41Z",
      "guid": "10360312-f33e-11e4-86c7-b8e85646e742",
      "zone_id": "1",
      "type": "ManageIQ::Providers::Redhat::InfraManager"
    }
  ]
}
```


- Create Provider
  - Request:
  - Response:

Create Provider

Request:

```
POST /api/providers
```

```
{
  "type"       : "ManageIQ::Providers::Redhat::InfraManager",
  "name"       : "rhevm102",
  "hostname"   : "rhevm102",
  "ipaddress" : "100.200.300.102",
  "credentials" : [
    {
      "userid"     : "default_userid",
      "password"   : "default_password"
    },
    {
      "userid"     : "metrics_userid",
      "password"   : "metrics_password",
      "auth_type" : "metrics"
    }
  ]
}
```

Response:

```
{
  "results": [
    {
      "id": "106",
      "href" : "http://localhost:3000/api/providers/106",
      "name": "rhevm102",
      "hostname": "rhevm102",
      "ipaddress": "100.200.300.102",
      "created_on": "2015-05-05T15:57:44Z",
      "updated_on": "2015-05-05T15:57:44Z",
      "guid": "acbd610e-f3f6-11e4-aaba-b8e85646e742",
      "zone_id": "1",
      "type": "ManageIQ::Providers::Redhat::InfraManager"
    }
  ]
}
```


- Update Provider
  - Request:
  - Response:

Update Provider

Request:

```
POST /api/providers/106
```

```
{
  "action"     : "edit",
  "ipaddress" : "100.200.300.112",
  "credentials" : [
    {
      "userid"     : "updated_metrics_userid",
      "password"   : "updated_metrics_password",
      "auth_type" : "metrics"
    }
```

```
  ]
}
```

Response:

```
{
  "href": "http://localhost:3000/api/providers/106",
  "id": "106",
  "name": "rhevm102",
  "hostname": "rhevm102",
  "ipaddress": "100.200.300.112",
  "created_on": "2015-05-06T13:49:11Z",
  "updated_on": "2015-05-06T13:53:06Z",
  "guid": "acbd610e-f3f6-11e4-aaba-b8e85646e742",
  "zone_id": "1",
  "type": "ManageIQ::Providers::Redhat::InfraManager"
}
```


- Refresh Provider
  - Request:
  - Response:

Refresh Provider

Request:

```
POST /api/providers/105
```

```
{
  "action" : "refresh"
}
```

Response:

```
{
  "success": true,
  "message": "Provider id:105 name:'sample_provider' refreshing",
  "href": "http://localhost:3000/api/providers/105",
  "task_id" : "23",
  "task_href" : "http://localhost:3000/api/tasks/23",
  "tasks" : [
    { "id" : "23", "href" : "http://localhost:3000/api/tasks/23"},
    { "id" : "24", "href" : "http://localhost:3000/api/tasks/24"}
  ]
}
```

Note: In the above example a multi-manager provider was refreshed and in the response, a task id and href pair is returned for each of the provider managers. This allows the caller to check the status of each manager refresh for completion.

- Delete Provider
  - Deleting a single Provider
    - Request:
    - Response:
    - Request:
    - Response:
  - Deleting multiple Providers
    - Request:
    - Response:

Delete Provider

Deleting a single Provider

Request:

```
DELETE /api/provider/105
```

Or via the delete action as follows:

```
POST /api/provider/105
```

```
{
  "action" : "delete"
}
```

Response:

```
{
  "success": true,
  "message": "Provider id:106 name:'rhevm102' deleting",
  "task_id": "625",
  "task_href": "http://localhost:3000/api/tasks/625",
  "href": "http://localhost:3000/api/providers/106"
}
```

Note:

Delete actions are done asynchronously as it could take a while to complete. The delete task can be queried if needed as follows

Request:

```
GET /api/tasks/625
```

Response:

```
{
  "href": "http://localhost:3000/api/tasks/625",
  "id": "625",
  "name": "Provider id:106 name:'rhevm102' deleting",
  "state": "Finished",
  "status": "Ok",
  "message": "Task completed successfully",
  "userid": "admin",
  "created_on": "2015-05-06T14:02:26Z",
  "updated_on": "2015-05-06T14:02:32Z"
}
```

Deleting multiple Providers

Request:

```
POST /api/providers

{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/providers/107" },
    { "href" : "http://localhost:3000/api/providers/108" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Provider id:107 name:'rhevm102' deleting",
      "task_id": "626",
      "task_href": "http://localhost:3000/api/tasks/626",
      "href": "http://localhost:3000/api/providers/107"
    },
    {
      "success": true,
      "message": "Provider id:108 name:'rhevm103' deleting",
      "task_id": "627",
      "task_href": "http://localhost:3000/api/tasks/627",
      "href": "http://localhost:3000/api/providers/108"
    }
  ]
}
```

Infrastructure management API Examples - Provisioning Requests

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Create a Provision Request
- Create multiple Provision Requests

- Trigger a single Provision Request
  - Request:
  - Response:

Trigger a single Provision Request

In the provision requests:

- version defaults to “1.1” if not specified.
- user_name defaults to the REST API authenticated user if not specified.

Request:

```
POST /api/provision_requests
```

```
{
  "version" : "1.1",
  "template_fields" : {
    "guid" : "afe6e8a0-89fd-11e3-b6ac-b8e85646e742"
  },
  "vm_fields" : {
    "number_of_cpus" : 1,
    "vm_name" : "aab_rest_vm1",
    "vm_memory" : "1024",
    "vlan" : "rhevm"
  },
  "requester" : {
    "user_name" : "jdoe",
    "owner_first_name" : "John",
    "owner_last_name" : "Doe",
    "owner_email" : "jdoe@sample.com",
    "auto_approve" : true
  },
  "tags" : {
    "network_location" : "Internal",
    "cc" : "001"
  },
  "additional_values" : {
    "request_id" : "1001"
  },
  "ems_custom_attributes" : { },
  "miq_custom_attributes" : { }
}
```

Optionally, the action based request format is also supported:

```
POST /api/provision_requests
```

```
{
  "action" : "create",
  "resource" : {
    "version" : "1.1",
      "template_fields" : {
        "guid" : "afe6e8a0-89fd-11e3-b6ac-b8e85646e742"
      },
      "vm_fields" : {
        "number_of_cpus" : 1,
        "vm_name" : "aab_rest_vm1",
        "vm_memory" : "1024",
        "vlan" : "rhevm"
      },
      "requester" : {
        "user_name" : "jdoe",
        "owner_first_name" : "John",
        "owner_last_name" : "Doe",
        "owner_email" : "jdoe@sample.com",
        "auto_approve" : true
      },
      "tags" : {
        "network_location" : "Internal",
        "cc" : "001"
      },
      "additional_values" : {
        "request_id" : "1001"
      },
      "ems_custom_attributes" : { },
      "miq_custom_attributes" : { }
  }
}
```

Response:


```
{
  "results": [
    {
      "href" : "http://localhost:3000/api/provision_requests/18",
      "id": "18",
      "description": "Provision from [bd-clone-template] to [aab_rest_vm1]",
      "approval_state": "pending_approval",
      "type": "MiqProvisionRequest",
      "created_on": "2015-05-08T17:42:55Z",
      "updated_on": "2015-05-08T17:42:55Z",
      "requester_id": "1",
      "requester_name": "Administrator",
      "request_type": "template",
      "request_state": "pending",
      "message": "VM Provisioning - Request Created",
      "status": "Ok",
      "options": {
        "use_pre_dialog": false,
        "request_type": "template",
        "miq_request_dialog_name": "miq_provision_redhat_dialogs_template",
        "owner_first_name": "John",
        "owner_last_name": "Doe",
        "owner_email": "jdoe@sample.com",
        "vm_tags": [
          62,
          58
        ],
        "addr_mode": [
          "static",
          "Static"
        ],
        "placement_cluster_name": [
          null,
          null
        ],
        "cluster_filter": [
          null,
          null
        ],
        "placement_auto": [
          true,
          1
        ],
        "placement_dc_name": [
          null,
          null
        ],
        "number_of_vms": [
          1,
          "1"
        ],
        "src_vm_id": [
          1947,
          "bd-clone-template"
        ],
        "provision_type": [
          "native_clone",
          "Native Clone"
        ],
        "linked_clone": [
          null,
          null
        ],
        "vm_name": "aab_rest_vm1",
        "pxe_server_id": [
          null,
          null
        ],
        "schedule_type": [
          "immediately",
          "Immediately on Approval"
        ],
        "vm_auto_start": [
          true,
          1
        ],
        "schedule_time": "2015-05-09T13:42:54-04:00",
        "retirement": [
          0,
```

```
          "Indefinite"
        ],
        "retirement_warn": [
          604800,
          "1 Week"
        ],
        "stateless": [
          false,
          0
        ],
        "vlan": [
          "rhevm",
          "rhevm"
        ],
        "disk_format": [
          "default",
          "Default"
        ],
        "number_of_sockets": [
          1,
          "1"
        ],
        "cores_per_socket": [
          1,
          "1"
        ],
        "vm_memory": [
          "1024",
          "1024"
        ],
        "network_adapters": [
          1,
          "1"
        ],
        "placement_host_name": [
          null,
          null
        ],
        "placement_ds_name": [
          null,
          null
        ],
        "src_vm_nics": [

        ],
        "src_vm_lans": [

        ],
        "customize_enabled": [
          "enabled"
        ],
        "src_ems_id": [
          105,
          "rhevm230"
        ],
        "auto_approve": false,
        "ws_values": {
          "request_id": "1001"
        },
        "ws_ems_custom_attributes": {
        },
        "ws_miq_custom_attributes": {
        }
      },
      "userid": "jdoe",
      "source_id": 1947,
      "source_type": "VmOrTemplate"
    }
  ]
}
```


- Trigger multiple Provision Requests
  - Request:
  - Response:

Trigger multiple Provision Requests

In the provision requests:

- version defaults to “1.1” if not specified.

- user_name defaults to the REST API authenticated user if not specified.

Request:

```
POST /api/provision_requests
```

```
{
  "action" : "create",
  "resources" : [
    {
      "version" : "1.1",
      "template_fields" : { "guid" : "afe6e8a0-89fd-11e3-b6ac-b8e85646e742" },
      "vm_fields" : {
        "vm_name" : "jdoe_rest_vm1",
        "number_of_cpus" : 1,
        "vm_memory" : "1024",
        "vlan" : "nic1"
      },
      "requester" : {
        "user_name" : "jdoe",
        "owner_first_name" : "John",
        "owner_last_name" : "Doe",
        "owner_email" : "jdoe@sample.com",
        "auto_approve" : true
      },
      "tags" : {
        "network_location" : "Internal",
        "cc" : "001"
      },
      "additional_values" : { "request_id" : "1001" },
      "ems_custom_attributes" : { },
      "miq_custom_attributes" : { }
    },
    {
      "template_fields" : { "guid" : "afe6e8a0-89fd-11e3-b6ac-b8e85646e742" },
      "vm_fields" : {
        "vm_name" : "jdoe_rest_vm2",
        "number_of_cpus" : 1,
        "vm_memory" : "2048",
        "vlan" : "nic1"
      },
      "requester" : {
        "owner_first_name" : "John",
        "owner_last_name" : "Doe",
        "owner_email" : "jdoe@sample.com",
        "auto_approve" : true
      },
      "tags" : {
        "network_location" : "Internal",
        "cc" : "001"
      },
      "additional_values" : { "request_id" : "1002" }
    },
    {
      "template_fields" : { "guid" : "afe6e8a0-89fd-11e3-b6ac-b8e85646e742" },
      "vm_fields" : {
        "vm_name" : "jdoe_rest_vm3",
        "number_of_cpus" : 1,
        "vm_memory" : "4096",
        "vlan" : "nic1"
      },
      "requester" : {
        "owner_first_name" : "John",
        "owner_last_name" : "Doe",
        "owner_email" : "jdoe@sample.com",
        "auto_approve" : true
      },
      "tags" : {
        "network_location" : "Internal",
        "cc" : "001"
      },
      "additional_values" : { "request_id" : "1003" }
    }
  ]
}
```

Response:

```
{
  "results": [
```

```
{
  "href": "http://localhost:3000/api/provision_requests/19",
  "id": "19",
  "description": "Provision from [bd-clone-template] to [aab_rest_vm1]",
  "approval_state": "pending_approval",
  "type": "MiqProvisionRequest",
  "created_on": "2015-05-08T18:25:25Z",
  "updated_on": "2015-05-08T18:25:26Z",
  "requester_id": "1",
  "requester_name": "jdoe",
  "request_type": "template",
  "request_state": "pending",
  "message": "VM Provisioning - Request Created",
  "status": "Ok",
  "options": {
    "use_pre_dialog": false,
    "request_type": "template",
    "miq_request_dialog_name": "miq_provision_redhat_dialogs_template",
    "owner_first_name": "John",
    "owner_last_name": "Doe",
    "owner_email": "jdoe@sample.com",
    "vm_tags": [
      62,
      58
    ],
    "addr_mode": [
      "static",
      "Static"
    ],
    "placement_cluster_name": [
      null,
      null
    ],
    "cluster_filter": [
      null,
      null
    ],
    "placement_auto": [
      true,
      1
    ],
    "placement_dc_name": [
      null,
      null
    ],
    "number_of_vms": [
      1,
      "1"
    ],
    "src_vm_id": [
      1947,
      "bd-clone-template"
    ],
    "provision_type": [
      "native_clone",
      "Native Clone"
    ],
    "linked_clone": [
      null,
      null
    ],
    "vm_name": "aab_rest_vm1",
    "pxe_server_id": [
      null,
      null
    ],
    "schedule_type": [
      "immediately",
      "Immediately on Approval"
    ],
    "vm_auto_start": [
      true,
      1
    ],
    "schedule_time": "2015-05-09T14:25:25-04:00",
    "retirement": [
      0,
      "Indefinite"
    ],
    "retirement_warn": [
      604800,
```

```
          "1 Week"
        ],
        "stateless": [
          false,
          0
        ],
        "vlan": [
          "rhevm",
          "rhevm"
        ],
        "disk_format": [
          "default",
          "Default"
        ],
        "number_of_sockets": [
          1,
          "1"
        ],
        "cores_per_socket": [
          1,
          "1"
        ],
        "vm_memory": [
          "1024",
          "1024"
        ],
        "network_adapters": [
          1,
          "1"
        ],
        "placement_host_name": [
          null,
          null
        ],
        "placement_ds_name": [
          null,
          null
        ],
        "src_vm_nics": [

        ],
        "src_vm_lans": [

        ],
        "customize_enabled": [
          "enabled"
        ],
        "src_ems_id": [
          105,
          "rhevm230"
        ],
        "auto_approve": false,
        "ws_values": {
          "request_id": "1001"
        },
        "ws_ems_custom_attributes": {
        },
        "ws_miq_custom_attributes": {
        }
      },
      "userid": "jdoe",
      "source_id": "1947",
      "source_type": "VmOrTemplate"
    },
    {
      "href": "http://localhost:3000/api/provision_requests/20",
      "id": "20",
      "description": "Provision from [bd-clone-template] to [aab_rest_vm2]",
      "approval_state": "pending_approval",
      "type": "MiqProvisionRequest",
      "created_on": "2015-05-08T18:25:28Z",
      "updated_on": "2015-05-08T18:25:29Z",
      "requester_id": "1",
      "requester_name": "jdoe",
      "request_type": "template",
      "request_state": "pending",
      "message": "VM Provisioning - Request Created",
      "status": "Ok",
      "options": {
        "use_pre_dialog": false,
        "request_type": "template",
```

```
        "miq_request_dialog_name": "miq_provision_redhat_dialogs_template",
        "owner_first_name": "John",
        "owner_last_name": "Doe",
        "owner_email": "jdoe@sample.com",
        "vm_tags": [
          62,
          58
        ],
        "addr_mode": [
          "static",
          "Static"
        ],
        "placement_cluster_name": [
          null,
          null
        ],
        "cluster_filter": [
          null,
          null
        ],
        "placement_auto": [
          true,
          1
        ],
        "placement_dc_name": [
          null,
          null
        ],
        "number_of_vms": [
          1,
          "1"
        ],
        "src_vm_id": [
          1947,
          "bd-clone-template"
        ],
        "provision_type": [
          "native_clone",
          "Native Clone"
        ],
        "linked_clone": [
          null,
          null
        ],
        "vm_name": "aab_rest_vm2",
        "pxe_server_id": [
          null,
          null
        ],
        "schedule_type": [
          "immediately",
          "Immediately on Approval"
        ],
        "vm_auto_start": [
          true,
          1
        ],
        "schedule_time": "2015-05-09T14:25:28-04:00",
        "retirement": [
          0,
          "Indefinite"
        ],
        "retirement_warn": [
          604800,
          "1 Week"
        ],
        "stateless": [
          false,
          0
        ],
        "vlan": [
          "rhevm",
          "rhevm"
        ],
        "disk_format": [
          "default",
          "Default"
        ],
        "number_of_sockets": [
          1,
          "1"
```

```
        ],
        "cores_per_socket": [
          1,
          "1"
        ],
        "vm_memory": [
          "1024",
          "1024"
        ],
        "network_adapters": [
          1,
          "1"
        ],
        "placement_host_name": [
          null,
          null
        ],
        "placement_ds_name": [
          null,
          null
        ],
        "src_vm_nics": [

        ],
        "src_vm_lans": [

        ],
        "customize_enabled": [
          "enabled"
        ],
        "src_ems_id": [
          105,
          "rhevm230"
        ],
        "auto_approve": false,
        "ws_values": {
          "request_id": "1001"
        },
        "ws_ems_custom_attributes": {
        },
        "ws_miq_custom_attributes": {
        }
      },
      "userid": "jdoe",
      "source_id": "1947",
      "source_type": "VmOrTemplate"
    },
    {
      "href": "http://localhost:3000/api/provision_requests/21",
      "id": "21",
      "description": "Provision from [bd-clone-template] to [aab_rest_vm3]",
      "approval_state": "pending_approval",
      "type": "MiqProvisionRequest",
      "created_on": "2015-05-08T18:25:32Z",
      "updated_on": "2015-05-08T18:25:32Z",
      "requester_id": "1",
      "requester_name": "jdoe",
      "request_type": "template",
      "request_state": "pending",
      "message": "VM Provisioning - Request Created",
      "status": "Ok",
      "options": {
        "use_pre_dialog": false,
        "request_type": "template",
        "miq_request_dialog_name": "miq_provision_redhat_dialogs_template",
        "owner_first_name": "John",
        "owner_last_name": "Doe",
        "owner_email": "jdoe@sample.com",
        "vm_tags": [
          62,
          58
        ],
        "addr_mode": [
          "static",
          "Static"
        ],
        "placement_cluster_name": [
          null,
          null
        ],
        "cluster_filter": [
```

```
          null,
          null
        ],
        "placement_auto": [
          true,
          1
        ],
        "placement_dc_name": [
          null,
          null
        ],
        "number_of_vms": [
          1,
          "1"
        ],
        "src_vm_id": [
          1947,
          "bd-clone-template"
        ],
        "provision_type": [
          "native_clone",
          "Native Clone"
        ],
        "linked_clone": [
          null,
          null
        ],
        "vm_name": "aab_rest_vm3",
        "pxe_server_id": [
          null,
          null
        ],
        "schedule_type": [
          "immediately",
          "Immediately on Approval"
        ],
        "vm_auto_start": [
          true,
          1
        ],
        "schedule_time": "2015-05-09T14:25:31-04:00",
        "retirement": [
          0,
          "Indefinite"
        ],
        "retirement_warn": [
          604800,
          "1 Week"
        ],
        "stateless": [
          false,
          0
        ],
        "vlan": [
          "rhevm",
          "rhevm"
        ],
        "disk_format": [
          "default",
          "Default"
        ],
        "number_of_sockets": [
          1,
          "1"
        ],
        "cores_per_socket": [
          1,
          "1"
        ],
        "vm_memory": [
          "1024",
          "1024"
        ],
        "network_adapters": [
          1,
          "1"
        ],
        "placement_host_name": [
          null,
          null
        ],
```

```
        "placement_ds_name": [
          null,
          null
        ],
        "src_vm_nics": [

        ],
        "src_vm_lans": [

        ],
        "customize_enabled": [
          "enabled"
        ],
        "src_ems_id": [
          105,
          "rhevm230"
        ],
        "auto_approve": false,
        "ws_values": {
          "request_id": "1001"
        },
        "ws_ems_custom_attributes": {
        },
        "ws_miq_custom_attributes": {
        }
      },
      "userid": "jdoe",
      "source_id": "1947",
      "source_type": "VmOrTemplate"
    }
  ]
}
```

Infrastructure management API Examples - Service Catalogs

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Add a Service Catalog
- Add multiple Service Catalogs
- Edit a Service Catalog
- Edit Service Catalogs
- Assign Service Templates
- Unassign Service Templates
- Delete Service Catalogs

- Adding a simple Service Catalog
  - Request:
  - Response:

Adding a simple Service Catalog

Request:

```
POST /api/service_catalogs
```

```
{
   "action" : "create",
   "resource" : {
        "name" : "Sample Service Catalog",
        "description" : "Description of Sample Service Catalog",
        "service_templates" : [
          { "href" : "http://localhost:3000/api/service_templates/3" },
          { "href" : "http://localhost:3000/api/service_templates/4" }
        ]
   }
}
```

Response:

```
{
  "results": [
    {
      "id": "7",
      "href" : "http://localhost:3000/api/service_catalogs/7",
```

```
      "name": "Sample Service Catalog",
      "description": "Description of Sample Service Catalog"
    }
  ]
}
```


- Adding multiple Service Catalogs
  - Request:
  - Response:

Adding multiple Service Catalogs

Request:

```
POST /api/service_catalogs
```

```
{
   "action" : "create",
   "resources" : [
      {
        "name" : "First Sample Service Catalog",
        "description" : "Description of First Sample Service Catalog",
        "service_templates" : [
          { "href" : "http://localhost:3000/api/service_templates/1" },
          { "href" : "http://localhost:3000/api/service_templates/2" }
        ]
      },
      {
        "name" : "Second Sample Service Catalog",
        "description" : "Description of Second Sample Service Catalog",
        "service_templates" : [
          { "href" : "http://localhost:3000/api/service_templates/3" },
          { "href" : "http://localhost:3000/api/service_templates/4" }
        ]
      },
      {
        "name" : "Third Sample Service Catalog",
        "description" : "Description of Third Sample Service Catalog",
        "service_templates" : [
          { "href" : "http://localhost:3000/api/service_templates/5" },
          { "href" : "http://localhost:3000/api/service_templates/6" }
        ]
      }
  ]
}
```

Response:

```
{
  "results": [
    {
      "id": "8",
      "href" : "http://localhost:3000/api/service_catalogs/8",
      "name": "First Sample Service Catalog",
      "description": "Description of First Sample Service Catalog"
    },
    {
      "id": "9",
      "href" : "http://localhost:3000/api/service_catalogs/9",
      "name": "Second Sample Service Catalog",
      "description": "Description of Second Sample Service Catalog"
    },
    {
      "id": "10",
      "href" : "http://localhost:3000/api/service_catalogs/10",
      "name": "Third Sample Service Catalog",
      "description": "Description of Third Sample Service Catalog"
    }
  ]
}
```


- Edit Service Catalog
  - Request:
  - Response:

Edit Service Catalog

Request:

```
POST /api/service_catalog/7

{
   "action" : "edit",
   "resource" : {
      "description" : "Updated Description of the Seventh Service Catalog"
   }
}
```

Response:

```
{
  "href": "http://localhost:3000/api/service_catalogs/7",
  "id": "7",
  "name": "Sample Service Catalog",
  "description": "Updated Description of the Seventh Service Catalog",
  "service_templates": {
    "count": 1,
    "resources": [
      {
        "href": "http://localhost:3000/api/service_catalogs/7/service_templates/3"
      }
    ]
  }
}
```


- Edit Multiple Service Catalogs
  - Request:
  - Response:

Edit Multiple Service Catalogs

Request:

```
POST /api/service_catalogs

{
   "action" : "edit",
   "resources" : [
     {
       "href" : "http://localhost:3000/api/service_catalogs/3",
       "description" : "Updated Description for Third Service Catalog"
     },
     {
       "href" : "http://localhost:3000/api/service_catalogs/6",
       "description" : "Updated Description for Sixth Service Catalog"
     }
   ]
}
```

Response:

```
{
  "results": [
    {
      "id" : "3",
      "href" : "http://localhost:3000/api/service_catalogs/3",
      "name": "Service Catalog B Added from REST API",
      "description": "Updated Description for Third Service Catalog"
    },
    {
      "id" : "6",
      "href" : "http://localhost:3000/api/service_catalogs/6",
      "name": "Service Catalog E Added from REST API",
      "description": "Updated Description for Sixth Service Catalog"
    }
  ]
}
```


- Assign service templates to Service Catalog 1
  - Request:
  - Response:

Assign service templates to Service Catalog 1

Request:

```
POST /api/service_catalogs/1/service_templates
```

```
{
  "action" : "assign",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_templates/5" },
    { "href" : "http://localhost:3000/api/service_templates/1" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Service Template id:5 name:'template5'",
      "service_template_id": "5",
      "service_template_href": "http://localhost:3000/api/service_templates/5",
      "href": "http://localhost:3000/api/service_catalogs/1"
    },
    {
      "success": false,
      "message": "Service Template 1 is currently assigned to Service Catalog 2",
      "service_template_id": "1",
      "service_template_href": "http://localhost:3000/api/service_templates/1",
      "href": "http://localhost:3000/api/service_catalogs/1"
    }
  ]
}
```


- Unassign service templates from Service Catalog 2
  - Request:
  - Response:

Unassign service templates from Service Catalog 2

Request:

```
POST /api/service_catalogs/6/service_templates
```

```
{
  "action" : "unassign",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_templates/1" },
    { "href" : "http://localhost:3000/api/service_templates/5" },
    { "href" : "http://localhost:3000/api/service_templates/8" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": false,
      "message": "Service Template 1 is not currently assigned to Service Catalog 3",
      "service_template_id": "1",
      "service_template_href": "http://localhost:3000/api/service_templates/1",
      "href": "http://localhost:3000/api/service_catalogs/6"
    },
    {
      "success": true,
      "message": "Unassigning Service Template id:5 name:'template5'",
      "service_template_id": "5",
      "service_template_href": "http://localhost:3000/api/service_templates/5",
      "href": "http://localhost:3000/api/service_catalogs/6"
    },
    {
      "success": false,
      "message": "Couldn't find ServiceTemplate with id=8",
      "href": "http://localhost:3000/api/service_catalogs/6"
    }
  ]
}
```

- Delete multiple service catalogs
  - Request:
  - Response:

Delete multiple service catalogs

Request:

```
POST /api/service_catalogs
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/service_catalogs/8"  },
    { "href" : "http://localhost:3000/api/service_catalogs/9"  },
    { "href" : "http://localhost:3000/api/service_catalogs/10" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "service_catalogs id: 8 deleting",
      "href": "http://localhost:3000/api/service_catalogs/8"
    },
    {
      "success": true,
      "message": "service_catalogs id: 9 deleting",
      "href": "http://localhost:3000/api/service_catalogs/9"
    },
    {
      "success": true,
      "message": "service_catalogs id: 10 deleting",
      "href": "http://localhost:3000/api/service_catalogs/10"
    }
  ]
}
```

Infrastructure management API Examples - Service Templates

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Edit a Service Template
- Edit Service Templates
- Delete Service Templates

- Edit Service Template
  - Request:
  - Response:

Edit Service Template

Request:

```
POST /api/service_templates/2
```

```
{
  "action" : "edit",
  "resource" : {
    "name" : "updated_svc_template_02",
    "description" : "This is an updated description for service template 02"
  }
}
```

Response:

```
{
  "href": "http://localhost:3000/api/service_templates/2",
  "id": "2",
```

```
  "name": "updated_svc_template_02",
  "description": "This is an updated description for service template 02",
  "guid": "6f7918b4-d6e7-11e4-9837-b8e85646e742",
  "options": {
  },
  "created_at": "2015-03-30T14:17:02Z",
  "updated_at": "2015-04-16T22:30:02Z",
  "service_type": "unknown",
  "service_template_catalog_id": "6"
}
```


- Edit Service Templates
  - Request:
  - Response:

Edit Service Templates

Request:

```
POST /api/service_templates
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/service_templates/1",
      "description" : "This is an updated description for the first sample service template"
    },
    {
      "href" : "http://localhost:3000/api/service_templates/2",
      "description" : "This is an updated description for the second sample service template"
    }
  ]
}
```

Response:

```
{
  "results": [
    {
      "id": "1",
      "name": "template1",
      "description": "This is an updated description for the first sample service template",
      "guid": "6a6fdf7e-d6e7-11e4-9837-b8e85646e742",
      "options": {
      },
      "created_at": "2015-03-30T14:16:53Z",
      "updated_at": "2015-04-16T22:33:09Z",
      "service_type": "unknown",
      "service_template_catalog_id": "3"
    },
    {
      "id": "2",
      "name": "updated_svc_template_02",
      "description": "This is an updated description for the second sample service template",
      "guid": "6f7918b4-d6e7-11e4-9837-b8e85646e742",
      "options": {
      },
      "created_at": "2015-03-30T14:17:02Z",
      "updated_at": "2015-04-16T22:33:09Z",
      "service_type": "unknown",
      "service_template_catalog_id": "6"
    }
  ]
}
```


- Delete multiple service templates
  - Request:
  - Response:

Delete multiple service templates

Request:

```
POST /api/service_templates
```

```
{
  "action" : "delete",
```

```
  "resources" : [
    { "href" : "http://localhost:3000/api/service_templates/4" },
    { "href" : "http://localhost:3000/api/service_templates/5" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "service_templates id: 4 deleting",
      "href": "http://localhost:3000/api/service_templates/4"
    },
    {
      "success": true,
      "message": "service_templates id: 5 deleting",
      "href": "http://localhost:3000/api/service_templates/5"
    }
  ]
}
```

Infrastructure management API Examples - Services

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Order a Service
- Order Multiple Services
- Edit a Service
- Edit a Service with Patch Method
- Edit a Service with PUT
- Edit Services
- Reconfigure a Service
- Refresh Service Template Dialog Fields
- Refresh Service Dialog Fields
- Retire a Service
- Retire Multiple Services
- Trigger a Custom Action on a Service
- Delete Services

- Order a single service from Service Catalog 1
  - Request:
  - Response:

Order a single service from Service Catalog 1

Request:

```
POST /api/service_catalogs/1/service_templates
```

```
{
  "action" : "order",
  "resource" : {
    "href" : "http://localhost:3000/api/service_templates/3",
    "option_0_vm_target_name" : "test-vm-0001",
    "option_0_vm_target_hostname" : "test-vm-0001"
  }
}
```

Response:

```
{
  results: [
    {
      "approval_state": "pending_approval",
      "created_on": "2014-07-02T19:28:12Z",
      "description": "Provisioning Service [aws-ubuntu-api] from [aws-ubuntu-api]",
      "destination_id": null,
      "destination_type": null,
      "fulfilled_on": null,
      "id": "13",
```

```
      "message": "Service_Template_Provisioning - Request Created",
      "options": {
        "dialog": {
          "dialog_option_0_vm_target_name" : "test-vm-0001",
          "dialog_option_0_vm_target_hostname" : "test-vm-0001"
        }
    }
    "request_state": "pending",
    "request_type": "clone_to_service",
    "requester_id": "1",
    "requester_name": "Administrator",
    "source_id": "6",
    "source_type": "ServiceTemplate",
    "status": "Ok",
    "updated_on": "2014-07-02T19:28:12Z",
    "userid": "admin"
    }
  ]
}
```


- Order multiple services from Service Catalog 2
  - Request:
  - Response:

Order multiple services from Service Catalog 2

Request:

```
POST /api/service_catalogs/2/service_templates
```

```
{
  "action" : "order",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/service_templates/3",
      "option_1_vm_target_name" : "sample-vm-1201",
      "option_2_vm_target_hostname" : "sample-vm-1201"
    },
    {
      "href" : "http://localhost:3000/api/service_templates/3",
      "option_1_vm_target_name" : "sample-vm-1202",
      "option_2_vm_target_hostname" : "sample-vm-1202"
    },
    {
      "href" : "http://localhost:3000/api/service_templates/4",
      "option_1_vm_target_name" : "dev-vm1",
      "option_2_vm_target_hostname" : "dev-vm1",
      "option_3_vm_memory" : '16384'
    },
  ]
}
```

Response:

```
{
  results: [
    {
      "approval_state": "pending_approval",
      "created_on": "2014-07-02T19:28:12Z",
      "description": "Provisioning Service [sample-vm-1201] from [sample-vm-1201]",
      "destination_id": null,
      "destination_type": null,
      "fulfilled_on": null,
      "id": "13",
      "message": "Service_Template_Provisioning - Request Created",
      "options": {
        "dialog": {
          "dialog_option_0_vm_target_name" : "test-vm-0001",
          "dialog_option_0_vm_target_hostname" : "test-vm-0001"
        }
    }
    "request_state": "pending",
    "request_type": "clone_to_service",
    "requester_id": "1",
    "requester_name": "Administrator",
    "source_id": "6",
    "source_type": "ServiceTemplate",
    "status": "Ok",
    "updated_on": "2014-07-02T19:28:12Z",
```

```
    "userid": "admin"
    },
    {
      "approval_state": "pending_approval",
      "created_on": "2014-07-02T19:28:12Z",
      "description": "Provisioning Service [sample-vm-1202] from [sample-vm-1202]",
      "destination_id": null,
      "destination_type": null,
      "fulfilled_on": null,
      "id": "13",
      "message": "Service_Template_Provisioning - Request Created",
      "options": {
        "dialog": {
          "dialog_option_0_vm_target_name" : "test-vm-0001",
          "dialog_option_0_vm_target_hostname" : "test-vm-0001"
        }
    }
    "request_state": "pending",
    "request_type": "clone_to_service",
    "requester_id": "1",
    "requester_name": "Administrator",
    "source_id": "6",
    "source_type": "ServiceTemplate",
    "status": "Ok",
    "updated_on": "2014-07-02T19:28:12Z",
    "userid": "admin"
    },
    {
      "approval_state": "pending_approval",
      "created_on": "2014-07-02T19:28:12Z",
      "description": "Provisioning Service [sample-vm-1201] from [sample-vm-1201]",
      "destination_id": null,
      "destination_type": null,
      "fulfilled_on": null,
      "id": "13",
      "message": "Service_Template_Provisioning - Request Created",
      "options": {
        "dialog": {
          "dialog_option_0_vm_target_name" : "test-vm-0001",
          "dialog_option_0_vm_target_hostname" : "test-vm-0001"
        }
    }
    "request_state": "pending",
    "request_type": "clone_to_service",
    "requester_id": "1",
    "requester_name": "Administrator",
    "source_id": "6",
    "source_type": "ServiceTemplate",
    "status": "Ok",
    "updated_on": "2014-07-02T19:28:12Z",
    "userid": "admin"
    }
  ]
}
```


- Edit Service
  - Request:
  - Response:

Edit Service

Request:

```
POST /api/services/101
```


```
{
  "action" : "edit",
  "resource" : {
    "name" : "service_101",
    "description" : "This is an updated description for service 101"
  }
}
```


Response:


```
{
  "href": "http://localhost:3000/api/services/101",
  "id": "101",
  "name": "service_101",
```

```
  "description": "This is an updated description for the service 101",
  "guid": "4a8f96de-a1a6-11e4-9f8d-b8e85646e742",
  "options": {
  },
  "created_at": "2015-01-21T19:47:11Z",
  "updated_at": "2015-04-16T22:17:15Z"
}
```


- Editing a resource via the Patch Method.
  - Request:
  - Response:

Editing a resource via the Patch Method.

Supported attribute actions for PATCH include, add, edit and remove.

Request:

```
PATCH /api/services/90
```

```
[
  { "action" : "edit", "path" : "name", "value" : "updated_service_90" },
  { "action" : "remove", "path" : "description"}
]
```

Response:

```
{
  "href": "http://localhost:3000/api/services/90",
  "id": "90",
  "name": "updated_service_90",
  "guid": "eb6fc61c-7c9b-11e4-8a3a-b8e85646e742",
  "options": {
  },
  "created_at": "2014-12-05T16:29:43Z",
  "updated_at": "2015-04-16T22:38:57Z"
}
```

Note:

Note that the description attribute is no longer defined for this service.

- Edit Service via PUT
  - Request:
  - Response:

Edit Service via PUT

Request:

```
PUT /api/services/90
```

```
{
  "name" : "new_service_90",
  "description" : "This is a new description for the service 90"
}
```

Response:

```
{
  "href": "http://localhost:3000/api/services/90",
  "id": "90",
  "name": "new_service_90",
  "description": "This is a new description for the service 90",
  "guid": "eb6fc61c-7c9b-11e4-8a3a-b8e85646e742",
  "options": {
  },
  "created_at": "2014-12-05T16:29:43Z",
  "updated_at": "2015-04-16T22:41:37Z"
}
```


- Edit Services
  - Request:
  - Response:

Edit Services

Request:

```
POST /api/services
```

```
{
  "action" : "edit",
  "resources" : [
    {
      "href" : "http://localhost:3000/api/services/81",
      "description" : "This is an updated description for service 81"
    },
    {
      "href" : "http://localhost:3000/api/services/82",
      "description" : "This is an updated description for service 82"
    }
  ]
}
```

Response:

```
{
  "results": [
    {
      "href": "http://localhost:3000/api/services/81",
      "id": "81",
      "name": "api_gen_A_81",
      "description": "This is an updated description for service 81",
      "guid": "eb6daaf8-7c9b-11e4-8a3a-b8e85646e742",
      "options": {
      },
      "created_at": "2014-12-05T16:29:43Z",
      "updated_at": "2015-04-16T22:43:37Z"
    },
    {
      "href": "http://localhost:3000/api/services/82",
      "id": "82",
      "name": "api_gen_A_82",
      "description": "This is an updated description for service 82",
      "guid": "eb6de400-7c9b-11e4-8a3a-b8e85646e742",
      "options": {
      },
      "created_at": "2014-12-05T16:29:43Z",
      "updated_at": "2015-04-16T22:43:37Z"
    }
  ]
}
```


- Reconfigure Service
  - Request:
  - Response:

Reconfigure Service

Request:

```
POST /api/services/2
```

```
{
  "action" : "reconfigure",
  "resource" : {
     "SampleText" : "default_12",
     "SampleHostIp" : "100.200.300.12"
  }
}
```

Response:

```
{
  "success": true,
  "message": "Service id:2 name:'SampleStaticServiceCatalogItem' reconfiguring",
  "result": {
    "id": "74",
    "description": "Service Reconfigure for: SampleStaticServiceCatalogItem",
    "approval_state": "pending_approval",
    "type": "ServiceReconfigureRequest",
    "created_on": "2016-06-10T20:34:13Z",
    "updated_on": "2016-06-10T20:34:13Z",
    "requester_id": "1",
```

```
    "requester_name": "Administrator",
    "request_type": "service_reconfigure",
    "request_state": "pending",
    "message": "Service Reconfigure - Request Created",
    "status": "Ok",
    "options": {
      "dialog": {
        "dialog_SampleText": "default_12",
        "dialog_SampleHostName": "",
        "dialog_SampleHostIp": "100.200.300.12",
        "dialog_sample_button_one": "complex_stuff",
        "dialog_vm_host_name": ""
      },
      "workflow_settings": {
        "resource_action_id": "23",
        "dialog_id": "3"
      },
      "src_id": "2",
      "requester_group": "EvmGroup-super_administrator"
    },
    "userid": "admin",
    "source_id": "2",
    "source_type": "Service",
    "tenant_id": "1",
    "service_order_id": "9",
    "process": true
  },
  "href": "http://localhost:3000/api/services/2"
}
```


- Refresh Dialog Fields
  - Request:
  - Response:

Refresh Dialog Fields

Refresh fields of a service template dialog:

Request:

```
POST /api/service_catalogs/1/service_templates
```


```
{
  "action" : "refresh_dialog_fields",
  "resource" : {
    "href" : "http://localhost:3000/api/service_templates/1",
    "dialog_fields" : {
      "my_field" : "current value for my_field"
    },
    "fields" : [ "my_field" ]
  }
}
```


Response:


```
{
  "results": [
    {
      "success": true,
      "message": "Refreshing dialog fields for Service Template id:1 name:'DynamicDialog'",
      "result": {
        "my_field": {
          "name": "my_field",
          "description": "my_field",
          "type": "DialogFieldDropDownList",
          "data_type": "string",
          "display": "edit",
          "display_method_options": {
          },
          "required": true,
          "required_method_options": {
          },
          "values": [
            [
              "",
              "Choose"
            ],
            [
              "6",
```

```
              "6"
            ],
            [
              "15",
              "15"
            ],
            [
              "2016-06-10T15:14:30Z",
              "2016-06-10T15:14:30Z"
            ]
          ],
          "values_method_options": {
          },
          "options": {
            "sort_by": "value",
            "sort_order": "ascending"
          },
          "label": "My Field",
          "position": 0,
          "dynamic": true,
          "show_refresh_button": true,
          "load_values_on_init": true,
          "read_only": false,
          "resource_action": {
            "resource_type": "DialogField",
            "ae_namespace": "Dialogs",
            "ae_class": "dynamic_dialogs",
            "ae_instance": "drop_down",
            "ae_attributes": {
            }
          }
        }
      },
      "service_template_id": "1",
      "service_template_href": "http://localhost:3000/api/service_templates/1",
      "href": "http://localhost:3000/api/service_catalogs/1"
    }
  ]
}
```


- Refresh Dialog Fields
  - Request:
  - Response:

Refresh Dialog Fields

Refresh fields of a service dialog:

Request:

```
POST /api/service_dialogs/2
```

```
{
  "action" : "refresh_dialog_fields",
  "resource" : {
    "dialog_fields" : {
      "my_field" : "current value for my_field"
    },
    "fields" : [ "my_field" ]
  }
}
```

Response:

```
{
  "success": true,
  "message": "Refreshing dialog fields for Service Dialog id:2 label:'Dynamic Drop-down test'",
  "result": {
    "my_field": {
      "name": "my_field",
      "description": "my_field",
      "type": "DialogFieldDropDownList",
      "data_type": "string",
      "display": "edit",
      "display_method_options": {
      },
      "required": true,
      "required_method_options": {
      },
```

```
      "values": [
        [
          "",
          "Choose"
        ],
        [
          "7",
          "7"
        ],
        [
          "13",
          "13"
        ],
        [
          "2016-06-10T17:59:36Z",
          "2016-06-10T17:59:36Z"
        ]
      ],
      "values_method_options": {
      },
      "options": {
        "sort_by": "value",
        "sort_order": "ascending"
      },
      "label": "My Field",
      "position": 0,
      "dynamic": true,
      "show_refresh_button": true,
      "load_values_on_init": true,
      "read_only": false,
      "resource_action": {
        "resource_type": "DialogField",
        "ae_namespace": "Dialogs",
        "ae_class": "dynamic_dialogs",
        "ae_instance": "drop_down",
        "ae_attributes": {
        }
      }
    }
  },
  "href": "http://localhost:3000/api/service_dialogs/2"
}
```


- Retire Service Immediately
  - Request:
  - Response:
  - Response:
- Retire Service In the Future
  - Request:
  - Response:

Retire Service Immediately

Request:

```
POST /api/services

{
  "action" : "request_retire",
  "resource" : { "href" : "http://localhost:3000/api/services/35" }
}
```

Response:

```
{
  "results" : [
    {
      "href": "http://localhost:3000/api/services/95",
      "id": "95",
      "name": "sample_service_95",
      "description": "Description for sample_service_95",
      "guid": "eb713538-7c9b-11e4-8a3a-b8e85646e742",
      "options": {
      },
      "created_at": "2014-12-05T16:29:43Z",
      "updated_at": "2015-05-08T19:32:00Z"
    }
```

```
  ]
}
```

Or simply targeting the service directly:

```
POST /api/services/95
```

```
{
  "action" : "request_retire"
}
```

Response:

```
{
  "href": "http://localhost:3000/api/services/95",
  "id": "95",
  "name": "sample_service_95",
  "description": "Description for sample_service_95",
  "guid": "eb713538-7c9b-11e4-8a3a-b8e85646e742",
  "options": {
  },
  "created_at": "2014-12-05T16:29:43Z",
  "updated_at": "2015-05-08T19:32:00Z"
}
```

Retire Service In the Future

Request:

```
POST /api/services/93
```

```
{
  "action" : "request_retire",
  "resource" : { "date" : "10/31/2015", "warn" : "7"  }
}
```

Response:

```
{
  "href": "http://localhost:3000/api/services/93",
  "id": "93",
  "name": "sample_service_93",
  "description": "Description for sample_service_93",
  "guid": "eb709e02-7c9b-11e4-8a3a-b8e85646e742",
  "options": {
  },
  "created_at": "2014-12-05T16:29:43Z",
  "updated_at": "2015-05-08T19:40:19Z",
  "retired": false,
  "retires_on": "2015-10-30",
  "retirement_warn": 7
}
```


- Retire Multiple Services
  - Request:
  - Response:

Retire Multiple Services

Request:

```
POST /api/services
```

```
{
  "action" : "request_retire",
  "resources" : [
      { "href" : "http://localhost:3000/api/services/100" },
      { "href" : "http://localhost:3000/api/services/101", "date" : "11/01/2015", "warn" : "4" },
      { "href" : "http://localhost:3000/api/services/102", "date" : "11/02/2015", "warn" : "4" }
  ]
}
```

Response:

```
{
  "results": [
    {
```

```
      "href": "http://localhost:3000/api/services/100",
      "id": "100",
      "name": "cloud_service_100",
      "description": "Description for cloud_service_100",
      "guid": "eb725f94-7c9b-11e4-8a3a-b8e85646e742",
      "options": {
      },
      "created_at": "2014-12-05T16:29:43Z",
      "updated_at": "2015-05-08T19:46:11Z"
    },
    {
      "href": "http://localhost:3000/api/services/101",
      "id": "101",
      "name": "cloud_service_101",
      "description": "Description for cloud_service_101",
      "guid": "4a8f96de-a1a6-11e4-9f8d-b8e85646e742",
      "options": {
      },
      "created_at": "2015-01-21T19:47:11Z",
      "updated_at": "2015-05-08T19:46:19Z",
      "retired": false,
      "retires_on": "2015-11-01",
      "retirement_warn": 4
    },
    {
      "href": "http://localhost:3000/api/services/102",
      "id": "102",
      "name": "cloud_service_102",
      "description": "Description for cloud_service_102",
      "guid": "e2da0cb0-e47e-11e4-a47f-b8e85646e742",
      "options": {
      },
      "created_at": "2015-04-16T21:23:55Z",
      "updated_at": "2015-05-08T19:46:19Z",
      "retired": false,
      "retires_on": "2015-11-02",
      "retirement_warn": 3
    }
  ]
}
```


- Trigger a Custom Action on a Service
  - Request:
  - Response:
  - Request:
  - Response:

Trigger a Custom Action on a Service

This page shows an ordered service that has a custom button with a dialog and how the custom action is triggered.

Here is the ordered service with the expanded custom actions:

Request:

```
GET /api/services/110?attributes=custom_actions
```

Response:


```
{
  "href": "http://localhost:3000/api/services/110",
  "id": "110",
  "name": "SampleServiceTemplate",
  "description": "Sample Service Template",
  "guid": "a3ab175c-8ca1-11e5-939f-000a27020067",
  "service_template_id": "9",
  "options": {
    "button_order": [
      "cb-10"
    ],
    "dialog": {
      "dialog_hostname": "myhostname",
      "dialog_ipaddress": "192.168.100.1",
      "dialog_environmenttype": "medium"
    }
  },
  "display": true,
  "created_at": "2015-11-16T20:35:56Z",
```

```
  "updated_at": "2015-11-16T20:35:56Z",
  "evm_owner_id": "1",
  "miq_group_id": "1",
  "tenant_id": "1",
  "custom_actions": {
    "buttons": [
      {
        "id": "10",
        "guid": "cdcec41c-8ca0-11e5-8c79-000a27020067",
        "description": "Query VM",
        "applies_to_class": "ServiceTemplate",
        "options": {
          "button_image": 4,
          "display": true
        },
        "userid": "admin",
        "created_on": "2015-11-16T20:29:58Z",
        "updated_on": "2015-11-16T20:29:58Z",
        "name": "QueryVM",
        "visibility": {
          "roles": [
            "_ALL_"
          ]
        },
        "applies_to_id": "9",
        "resource_action": {
          "id": "38",
          "dialog_id": "4",
          "resource_id": "10",
          "resource_type": "CustomButton",
          "created_at": "2015-11-16T20:29:58Z",
          "updated_at": "2015-11-16T20:29:58Z",
          "ae_namespace": "SYSTEM",
          "ae_class": "PROCESS",
          "ae_instance": "Request",
          "ae_attributes": {
            "attr1": "value1",
            "attr2": "value2",
            "request": "System/Project/InspectME"
          },
          "dialog": {
            "description": "Sample Dialog Description",
            "buttons": "submit",
            "label": "SampleDialog",
            "dialog_tabs": [
              {
                "description": "VM Tab Description",
                "display": "edit",
                "label": "VMTabLabel",
                "position": 0,
                "dialog_groups": [
                  {
                    "description": "Vm Info Description",
                    "display": "edit",
                    "label": "VmInfoLabel",
                    "position": 0,
                    "dialog_fields": [
                      {
                        "name": "hostname",
                        "description": "Hostname of the VM",
                        "type": "DialogFieldTextBox",
                        "display": "edit",
                        "display_method_options": {
                        },
                        "required": false,
                        "required_method_options": {
                        },
                        "default_value": "default_hostname",
                        "values_method_options": {
                        },
                        "options": {
                          "protected": false
                        },
                        "label": "HostNameLabel",
                        "position": 0,
                        "dynamic": false,
                        "read_only": false,
                        "resource_action": {
                          "resource_type": "DialogField",
                          "ae_attributes": {
                          }
```

```
                    }
                  },
                  {
                    "name": "ipaddress",
                    "description": "IP Address of VM",
                    "type": "DialogFieldTextBox",
                    "display": "edit",
                    "display_method_options": {
                    },
                    "required": false,
                    "required_method_options": {
                    },
                    "default_value": "",
                    "values_method_options": {
                    },
                    "options": {
                      "protected": false
                    },
                    "label": "IpaddressLabel",
                    "position": 1,
                    "dynamic": false,
                    "read_only": false,
                    "resource_action": {
                      "resource_type": "DialogField",
                      "ae_attributes": {
                      }
                    }
                  }
                ]
              }
            ]
          },
          {
            "description": "Environment Tab Description",
            "display": "edit",
            "label": "EnvironmentTab",
            "position": 1,
            "dialog_groups": [
              {
                "description": "Environment Box Description",
                "display": "edit",
                "label": "EnvironmentBoxLabel",
                "position": 0,
                "dialog_fields": [
                  {
                    "name": "environmenttype",
                    "description": "Environment Type Description",
                    "type": "DialogFieldRadioButton",
                    "data_type": "string",
                    "display": "edit",
                    "display_method_options": {
                    },
                    "required": true,
                    "required_method_options": {
                    },
                    "values": [
                      [
                        "large",
                        "Large Environment"
                      ],
                      [
                        "medium",
                        "Medium Environment"
                      ],
                      [
                        "small",
                        "Small Environment"
                      ]
                    ],
                    "values_method_options": {
                    },
                    "options": {
                      "sort_by": "description",
                      "sort_order": "ascending"
                    },
                    "label": "EnvTypeLabel",
                    "position": 0,
                    "dynamic": false,
                    "read_only": false,
                    "resource_action": {
                      "resource_type": "DialogField",
```

```
                          "ae_attributes": {
                          }
                        }
                      }
                    ]
                  }
                ]
              }
            ]
          }
        }
      }
    ],
    "button_groups": [
```

```
    ]
  },
  "actions": [
    {
      "name": "edit",
      "method": "post",
      "href": "http://localhost:3000/api/services/110"
    },
    {
      "name": "request_retire",
      "method": "post",
      "href": "http://localhost:3000/api/services/110"
    },
    {
      "name": "set_ownership",
      "method": "post",
      "href": "http://localhost:3000/api/services/110"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "http://localhost:3000/api/services/110"
    },
    {
      "name": "queryvm",
      "method": "post",
      "href": "http://localhost:3000/api/services/110"
    }
  ]
}
```

Showing the **queryvm** custom action being returned in the actions list.

The following is the request triggering a **queryvm** action on the service:

Request:

```
POST /api/services/110
```

```
{
  "action" : "queryvm",
  "resource" : {
    "hostname": "updated_hostname",
    "ipaddress": "192.168.100.2",
    "environmenttype" : "medium"
  }
}
```

Response:

```
{
  "success": true,
  "message": "Invoked custom dialog action queryvm for services id: 110",
  "href": "http://localhost:3000/api/services/110"
}
```


- Delete Services
  - Request:
  - Response:

Delete Services

Request:

```
POST /api/services
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/services/97" },
    { "href" : "http://localhost:3000/api/services/98" },
    { "href" : "http://localhost:3000/api/services/99" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "services id: 97 deleting",
      "href": "http://localhost:3000/api/services/97"
    },
    {
      "success": true,
      "message": "services id: 98 deleting",
      "href": "http://localhost:3000/api/services/98"
    },
    {
      "success": true,
      "message": "services id: 99 deleting",
      "href": "http://localhost:3000/api/services/99"
    }
  ]
}
```

Infrastructure management API Examples - VM Management

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Start a VM
- Stop a VM
- Suspend a VM
- Scan a VM
- Set owner of a VM
- Add Event to a VM
- Delete a VM

- Start a VM
  - Request:
  - Response:
- Querying task progress:
  - Request:
  - Response:

Start a VM

Request:

```
POST /api/vms/1921
```

```
{
  "action": "start"
}
```

Response:

```
{
  "success": true,
  "message": "VM id:1921 name:'aab_demo_vm' starting",
  "task_id": "610",
  "task_href": "http://localhost:3000/api/tasks/610",
  "href": "http://localhost:3000/api/vms/1921"
}
```

Querying task progress:

Request:

```
GET /api/tasks/620
```

Response:

```
{
  "href": "http://localhost:3000/api/tasks/610",
  "id": "610",
  "name": "VM id:1921 name:'aab_demo_vm' starting",
  "state": "Queued",
  "status": "Ok",
  "message": "Queued the action: [VM id:1921 name:'aab_demo_vm' starting] being run for user: [admin]",
  "userid": "admin",
  "created_on": "2015-05-05T15:58:08Z",
  "updated_on": "2015-05-05T15:58:08Z"
}
```


- Stop a VM
  - Request:
  - Response:

Stop a VM

Request:

```
POST /api/vms/1921
```

```
{
  "action": "stop"
}
```

Response:

```
{
  "success": true,
  "message": "VM id:1921 name:'aab_demo_vm' stopping",
  "task_id": "619",
  "task_href": "http://localhost:3000/api/tasks/619",
  "href": "http://localhost:3000/api/vms/1921"
}
```


- Suspend a VM
  - Request:
  - Response:

Suspend a VM

Request:

```
POST /api/vms/1921
```

```
{
  "action": "suspend"
}
```

Response:

```
{
  "success": true,
  "message": "VM id:1921 name:'aab_demo_vm' suspending",
  "task_id": "620",
  "task_href": "http://localhost:3000/api/tasks/620",
  "href": "http://localhost:3000/api/vms/1921"
}
```


- Scan a VM
  - Request:
  - Response:
  - Response:

Scan a VM

Request:

```
POST /api/vms/1922

{
  "action": "scan"
}
```

Response:

```
{
  "success": true,
  "message": "VM id:1922 name:'aab_test_vm' scanning",
  "task_id": "618",
  "task_href": "http://localhost:3000/api/tasks/618",
  "href": "http://localhost:3000/api/vms/1922"
}
```

And then optionally waiting for the task to complete as follows:

```
GET /api/tasks/618
```

Response:

```
{
  "href": "http://localhost:3000/api/tasks/618",
  "id": "618",
  "name": "VM id:1922 name:'aab_test_vm' scanning",
  "state": "Finished",
  "status": "Ok",
  "message": "Task completed successfully",
  "userid": "admin",
  "created_on": "2015-05-05T19:37:32Z",
  "updated_on": "2015-05-05T19:37:38Z"
}
```


- Set owner of a VM
  - Request:
  - Response:

Set owner of a VM

Request:

```
POST /api/vms/1921

{
  "action": "set_owner",
  "resource" : {
    "owner" : "admin"
  }
}
```

Response:

```
{
  "success": true,
  "message": "VM id:1921 name:'aab_demo_vm' setting owner to 'admin'",
  "href": "http://localhost:3000/api/vms/1921"
}
```


- Add event to a VM
  - Request:
  - Response:

Add event to a VM

Request:

```
POST /api/vms/1921

{
  "action": "add_event",
  "resource" : {
    "event_type" : "BadUserNameSessionEvent",
    "event_message" : "Cannot login user@test.domain"
```

```
  }
}
```

Response:

```
{
  "success": true,
  "message": "Adding Event type=BadUserNameSessionEvent message=Cannot login user@test.domain",
  "href": "http://localhost:3000/api/vms/1921"
}
```


- Delete VMs
  - Deleting a single VM
    - Request:
  - Deleting multiple VMs
    - Request:
    - Response:
    - Request:
    - Response:

Delete VMs

Deleting a single VM

Request:

```
DELETE /api/vms/334
```

Upon success, Vm targeted for deletion asynchronously and No Response Content with HTTP status code 204

Deleting multiple VMs

Request:

```
DELETE /api/vms
```

```
{
  "action" : "delete",
  "resources" : [
    { "href" : "http://localhost:3000/api/vms/348" },
    { "href" : "http://localhost:3000/api/vms/349" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "VM id:348 name:'aab-temp1' deleting",
      "task_id": "616",
      "task_href": "http://localhost:3000/api/tasks/616",
      "href": "http://localhost:3000/api/vms/348"
    },
    {
      "success": true,
      "message": "VM id:349 name:'aab-temp2' deleting",
      "task_id": "617",
      "task_href": "http://localhost:3000/api/tasks/617",
      "href": "http://localhost:3000/api/vms/349"
    }
  ]
}
```

Optionally monitoring the asynchronous Vm deletion by accessing the related Task as follows:

Request:

```
GET /api/tasks/616
```

Response:

```
{
  "href": "http://localhost:3000/api/tasks/616",
```

```
  "id": "616",
  "name": "VM id:348 name:'aab-temp1' deleting",
  "state": "Finished",
  "status": "Ok",
  "message": "Task completed successfully",
  "userid": "admin",
  "created_on": "2015-05-05T19:33:35Z",
  "updated_on": "2015-05-05T19:33:40Z"
}
```

Infrastructure management API Examples - Tag Management

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Assign Tags
- Assign Tags to Service Template
- Unassign Tags
- Unassign Tags from Service Template

- Assign Tags to a Service
  - Request:
  - Response:
- Assign Tags by Name to a Service
  - Request:
  - Response:
- Assign a Tag by Reference to a Service
  - Request:
  - Response:

Assign Tags to a Service

Request:

```
POST /api/services/101/tags
```

```
{
  "action" : "assign",
  "resources" : [
    { "category" : "location", "name" : "ny" },
    { "category" : "department", "name" : "finance" },
    { "category" : "environment", "name" : "dev" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Tag: category:'location' name:'ny'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "location",
      "tag_name": "ny"
    },
    {
      "success": true,
      "message": "Assigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "department",
      "tag_name": "finance"
    },
    {
      "success": true,
      "message": "Assigning Tag: category:'environment' name:'dev'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "environment",
      "tag_name": "dev"
    }
  ]
}
```

Assign Tags by Name to a Service

Request:

```
POST /api/services/101/tags
```

```
{
  "action" : "assign",
  "resources" : [
    { "name" : "/department/finance" },
    { "name" : "/location/ny" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "department",
      "tag_name": "finance"
    },
    {
      "success": true,
      "message": "Assigning Tag: category:'location' name:'ny'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "location",
      "tag_name": "ny"
    }
  ]
}
```

Assign a Tag by Reference to a Service

Request:

```
POST /api/services/101/tags
```

```
{
  "action" : "assign",
  "resources" : [
    { "href" : "http://localhost:3000/api/services/1/tags/49" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "department",
      "tag_name": "finance",
      "tag_href": "http://localhost:3000/api/tags/49"
    }
  ]
}
```


- Assign Tags to service template 1
  - Request:
  - Response:

Assign Tags to service template 1

Request:

```
POST /api/service_templates/1/tags
```

```
{
  "action" : "assign",
  "resources" : [
    { "category" : "location", "name" : "ny" },
```

```
    { "category" : "department", "name" : "finance" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Tag: category:'location' name:'ny'",
      "href": "http://localhost:3000/api/service_templates/1",
      "tag_category": "location",
      "tag_name": "ny"
    },
    {
      "success": true,
      "message": "Assigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/service_templates/1",
      "tag_category": "department",
      "tag_name": "finance"
    }
  ]
}
```


- Unassign Tags from a Service
  - Request:
  - Response:
- Unassign a Tag by Name from a Service
  - Request:
  - Response:
- Unassign a Tag by Reference from a Service
  - Request:
  - Response:

Unassign Tags from a Service

Request:

```
POST /api/services/101/tags
```

```
{
  "action" : "unassign",
  "resources" : [
    { "category" : "department", "name" : "finance" },
    { "category" : "environment", "name" : "dev" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Unassigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "department",
      "tag_name": "finance"
    },
    {
      "success": true,
      "message": "Unassigning Tag: category:'environment' name:'dev'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "environment",
      "tag_name": "dev"
    }
  ]
}
```

Unassign a Tag by Name from a Service

Request:

```
POST /api/services/101/tags
```

```
{
  "action" : "unassign",
  "resources" : [
    { "name" : "/managed/department/finance" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Unassigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "department",
      "tag_name": "finance"
    }
  ]
}
```

Unassign a Tag by Reference from a Service

Request:

```
POST /api/services/101/tags
```

```
{
  "action" : "unassign",
  "resources" : [
    { "href" : "tags/49" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Unassigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/services/101",
      "tag_category": "department",
      "tag_name": "finance",
      "tag_href": "http://localhost:3000/api/tags/49"
    }
  ]
}
```


- Unassign Tags from service template 1
  - Request:
  - Response:

Unassign Tags from service template 1

Request:

```
POST /api/service_templates/1/tags
```

```
{
  "action" : "unassign",
  "resources" : [
    { "category" : "location", "name" : "ny" },
    { "category" : "department", "name" : "finance" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Unassigning Tag: category:'location' name:'ny'",
      "href": "http://localhost:3000/api/service_templates/1",
      "tag_category": "location",
      "tag_name": "ny"
```

```
    },
    {
      "success": true,
      "message": "Unassigning Tag: category:'department' name:'finance'",
      "href": "http://localhost:3000/api/service_templates/1",
      "tag_category": "department",
      "tag_name": "finance"
    }
  ]
}
```

Infrastructure management API Examples - Usage Flows

In this guide you will find examples for the Infrastructure Automation – Infrastructure Management REST API.

- Compliance Policy Flow
- Compliance Policy Flow
  - Create a couple of conditions for the new compliance policy
    - Request:
    - Response:
  - Find the Policy action to use
    - Request:
    - Response:
  - Find the Policy event to use
    - Response:
  - Create the Compliance Policy for Finance Servers
    - Request:
    - Response:
  - Create the Compliance Policy Profile for Finance Servers
    - Request:
    - Response:
  - Assigning new policy to policy profile
    - Request:
    - Response:
  - Assigning the policy profile to a virtual machine
    - Request:
    - Response:
  - Querying the policy profiles and related policies of the virtual machine
    - Request:
    - Response:
  - Requesting a compliance check on the virtual machine
    - Request:
    - Response:
  - Waiting for the compliance check to complete
    - Response:
  - Querying all compliances run for the virtual machine
    - Request:
    - Response:
  - Querying the virtual machine and its latest compliance result including the last compliance status
    - Request:
    - Response:

Compliance Policy Flow

Create a couple of conditions for the new compliance policy

Request:

```
POST /api/conditions
```

```
{
  "action" : "create",
  "resources" : [
    {
```

```
      "name" : "finance_mem_req",
      "description" : "System Memory Requirement for Finance Systems",
      "expression" : {
        ">=": {
          "field": "Vm-mem_cpu",
          "value": "4096"
        }
      },
      "towhat" : "Vm"
    },
    {
      "name" : "finance_cpu_req",
      "description" : "System CPU Requirement for Finance Systems",
      "expression" : {
        ">=": {
          "field": "Vm-num_cpu",
          "value": "2"
        }
      },
      "towhat" : "Vm"
    }
  ]
}
```

Response:

```
{
  "results": [
    {
      "href": "http://localhost:3000/api/conditions/8",
      "id": "8",
      "name": "finance_mem_req",
      "description": "System Memory Requirement for Finance Systems",
      "expression": {
        "exp": {
          ">=": {
            "field": "Vm-mem_cpu",
            "value": "4096"
          }
        },
        "context_type": null
      },
      "towhat": "Vm",
      "file_mtime": null,
      "created_on": "2020-03-24T17:19:21Z",
      "updated_on": "2020-03-24T17:19:21Z",
      "guid": "b67a564d-5290-4245-b4bc-4d75d0890951",
      "filename": null,
      "applies_to_exp": null,
      "miq_policy_id": null,
      "notes": null,
      "read_only": null
    },
    {
      "href": "http://localhost:3000/api/conditions/9",
      "id": "9",
      "name": "finance_cpu_req",
      "description": "System CPU Requirement for Finance Systems",
      "expression": {
        "exp": {
          ">=": {
            "field": "Vm-num_cpu",
            "value": "2"
          }
        },
        "context_type": null
      },
      "towhat": "Vm",
      "file_mtime": null,
      "created_on": "2020-03-24T17:19:21Z",
      "updated_on": "2020-03-24T17:19:21Z",
      "guid": "2c3c6fd1-c6a4-44e7-8a86-b1f44c1e2572",
      "filename": null,
      "applies_to_exp": null,
      "miq_policy_id": null,
      "notes": null,
      "read_only": null
    }
  ]
}
```

Find the Policy action to use

Request:

```
GET /api/policy_actions?expand=resources&filter[]=name=compliance_failed
```

Response:


```
{
  "name": "policy_actions",
  "count": 42,
  "subcount": 1,
  "subquery_count": 1,
  "pages": 1,
  "resources": [
    {
      "href": "http://localhost:3000/api/policy_actions/34",
      "id": "34",
      "name": "compliance_failed",
      "description": "Mark as Non-Compliant",
      "created_on": "2019-08-08T19:40:10Z",
      "updated_on": "2019-08-08T19:40:10Z",
      "guid": "47969a3f-4eb8-4e7b-b7d2-542e4c0cb0d5",
      "action_type": "default",
      "options": {
      }
    }
  ],
  "actions": [
    {
      "name": "query",
      "method": "post",
      "href": "http://localhost:3000/api/policy_actions"
    }
  ],
  "links": {
    "self": "http://localhost:3000/api/policy_actions?
attributes=name%2Cname&expand=resources&filter%5B%5D=name%3Dcompliance_failed&offset=0",
    "first": "http://localhost:3000/api/policy_actions?
attributes=name%2Cname&expand=resources&filter%5B%5D=name%3Dcompliance_failed&offset=0",
    "last": "http://localhost:3000/api/policy_actions?
attributes=name%2Cname&expand=resources&filter%5B%5D=name%3Dcompliance_failed&offset=0"
  }
}
```


Find the Policy event to use

```
GET /api/events?expand=resources&filter[]=name=vm_compliance_failed
```

Response:


```
{
  "name": "events",
  "count": 191,
  "subcount": 1,
  "subquery_count": 1,
  "pages": 1,
  "resources": [
    {
      "href": "http://localhost:3000/api/events/119",
      "id": "119",
      "name": "vm_compliance_check",
      "description": "VM Compliance Check",
      "created_on": "2019-08-08T19:40:12Z",
      "updated_on": "2019-08-08T19:40:12Z",
      "guid": "f95fd930-b70d-4198-b018-7f2fe37a33f9",
      "event_type": "Default",
      "definition": null,
      "default": null,
      "enabled": null
    }
  ],
  "actions": [
    {
      "name": "query",
      "method": "post",
      "href": "http://localhost:3000/api/events"
    }
  ],
```

```
  "links": {
    "self": "http://localhost:3000/api/events?
expand=resources&filter%5B%5D=name%3Dvm_compliance_check&offset=0",
    "first": "http://localhost:3000/api/events?
expand=resources&filter%5B%5D=name%3Dvm_compliance_check&offset=0",
    "last": "http://localhost:3000/api/events?
expand=resources&filter%5B%5D=name%3Dvm_compliance_check&offset=0"
  }
}
```

Create the Compliance Policy for Finance Servers

Request:

```
POST /api/policies
```

```
{
  "name" : "finance_server_policy",
  "description" : "Finance Server Policy",
  "mode" : "compliance",
  "towhat" : "Vm",
  "conditions_ids" : [ "8", "9" ],
  "policy_contents" : [
    {
      "event_id" : "119",
      "actions" : [
        {
          "action_id" : "34",
          "opts" : { "qualifier" : "failure" }
        }
      ]
    }
  ]
}
```

Response:

```
{
  "results": [
    {
      "href": "http://localhost:3000/api/policies/11",
      "id": "11",
      "name": "finance_server_policy",
      "description": "Finance Server Policy",
      "created_on": "2020-03-25T16:25:06Z",
      "updated_on": "2020-03-25T16:25:06Z",
      "expression": null,
      "towhat": "Vm",
      "guid": "95f28050-7947-41e4-8082-ee9cd512b5fd",
      "created_by": null,
      "updated_by": null,
      "notes": null,
      "active": true,
      "mode": "compliance",
      "read_only": null
    }
  ]
}
```

Create the Compliance Policy Profile for Finance Servers

Request:

```
POST /api/policy_profiles
```

```
{
  "name" : "finance_policy_profile",
  "description" : "Finance Servers Policy Profile",
  "mode" : "compliance"
}
```

Response:

```
{
  "results": [
    {
      "href": "http://localhost:3000/api/policy_profiles/35",
      "id": "35",
```

```
      "name": "finance_policy_profile",
      "description": "Finance Servers Policy Profile",
      "set_type": "MiqPolicySet",
      "created_on": "2020-03-24T18:56:23Z",
      "updated_on": "2020-03-24T18:56:23Z",
      "guid": "d4546f30-15a7-48f3-a793-cfd6d1d849a2",
      "read_only": null,
      "set_data": null,
      "mode": "compliance",
      "owner_type": null,
      "owner_id": null,
      "userid": null,
      "group_id": null
    }
  ]
}
```

Assigning new policy to policy profile

Request:

```
POST /api/policy_profiles/35/policies
```

```
{
  "action" : "assign",
  "resources" : [
    { "id" : "11" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Policy: id:'11' description:'Finance Server Policy' guid:'95f28050-7947-41e4-
8082-ee9cd512b5fd'",
      "href": "http://localhost:3000/api/policy_profiles/35",
      "policy_id": "12",
      "policy_href": "http://localhost:3000/api/policies/11"
    }
  ]
}
```

Assigning the policy profile to a virtual machine

Request:

```
POST /api/vms/171/policy_profiles
```

```
{
  "action" : "assign",
  "resources" : [
    { "id" : "35" }
  ]
}
```

Response:

```
{
  "results": [
    {
      "success": true,
      "message": "Assigning Policy Profile: id:'35' description:'Finance Servers Policy Profile'
guid:'d4546f30-15a7-48f3-a793-cfd6d1d849a2'",
      "href": "http://localhost:3000/api/vms/171",
      "policy_profile_id": "35",
      "policy_profile_href": "http://localhost:3000/api/policy_profiles/35"
    }
  ]
}
```

Querying the policy profiles and related policies of the virtual machine

Request:

```
GET /api/vms/171/policy_profiles?expand=resources&attributes=miq_policies
```

Response:

```
{
  "name": "policy_profiles",
  "count": 6,
  "subcount": 1,
  "pages": 1,
  "resources": [
    {
      "href": "http://localhost:3000/api/vms/171/policy_profiles/35",
      "id": "35",
      "name": "finance_policy_profile",
      "description": "Finance Servers Policy Profile",
      "set_type": "MiqPolicySet",
      "created_on": "2020-03-24T18:56:23Z",
      "updated_on": "2020-03-24T18:56:23Z",
      "guid": "d4546f30-15a7-48f3-a793-cfd6d1d849a2",
      "read_only": null,
      "set_data": null,
      "mode": "compliance",
      "owner_type": null,
      "owner_id": null,
      "userid": null,
      "group_id": null,
      "miq_policies": [
        {
          "href": "http://localhost:3000/api/policies/12",
          "id": "12",
          "name": "finance_server_policy",
          "description": "Finance Server Policy",
          "created_on": "2020-03-25T16:44:19Z",
          "updated_on": "2020-03-25T16:44:19Z",
          "expression": null,
          "towhat": "Vm",
          "guid": "e56d468c-1b1b-42f1-a4cb-4ea11924d7b3",
          "created_by": null,
          "updated_by": null,
          "notes": null,
          "active": true,
          "mode": "compliance",
          "read_only": null
        }
      ]
    }
  ],
  "actions": [
    {
      "name": "assign",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171/policy_profiles"
    },
    {
      "name": "unassign",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171/policy_profiles"
    },
    {
      "name": "resolve",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171/policy_profiles"
    }
  ],
  "links": {
    "self": "http://localhost:3000/api/vms/171/policy_profiles?
attributes=miq_policies&expand=resources&offset=0",
    "first": "http://localhost:3000/api/vms/171/policy_profiles?
attributes=miq_policies&expand=resources&offset=0",
    "last": "http://localhost:3000/api/vms/171/policy_profiles?
attributes=miq_policies&expand=resources&offset=0"
  }
}
```

At this point, the compliance policy is properly assigned to the virtual machine. If configured via a schedule for the encompassing region, the compliance checks will run as scheduled. They can also be executed on the specific virtual machine via the API as follows:

Requesting a compliance check on the virtual machine

Request:

```
POST /api/vms/171

{
  "action" : "check_compliance"
}
```

Response:

```
{
  "success" : true,
  "message": "VM id:171 name:'aab-ldap' check compliance requested",
  "task_id": "35",
  "task_href": "http://localhost:3000/api/tasks/41",
  "href": "http://localhost:3000/api/vms/171"
}
```

Waiting for the compliance check to complete

One can find out when the compliance check completes by querying the task generated for it and look for the state attribute as follows:

```
GET /api/tasks/41
```

Response:

```
{
  "href": "http://localhost:3000/api/tasks/41",
  "id": "41",
  "name": "VM id:171 name:'aab-ldap' check compliance requested",
  "state": "Finished",
  "status": "Ok",
  "message": "Task completed successfully",
  "userid": "admin",
  "created_on": "2020-04-09T21:16:07Z",
  "updated_on": "2020-04-09T21:16:12Z",
  "pct_complete": null,
  "context_data": null,
  "results": null,
  "miq_server_id": "3",
  "identifier": null,
  "started_on": "2020-04-09T21:16:09Z",
  "zone": null,
  "actions": [
    {
      "name": "delete",
      "method": "post",
      "href": "http://localhost:3000/api/tasks/41"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "http://localhost:3000/api/tasks/41"
    }
  ]
}
```

Querying all compliances run for the virtual machine

Request:

```
GET /api/vms/171/compliances?expand=resources
```

Response:

```
{
  "name": "compliances",
  "count": 430,
  "subcount": 2,
  "pages": 1,
  "resources": [
    {
      "href": "http://localhost:3000/api/vms/171/compliances/40",
      "id": "40",
      "resource_id": "171",
      "resource_type": "VmOrTemplate",
      "compliant": false,
      "timestamp": "2020-03-26T14:30:24Z",
```

```
      "updated_on": "2020-03-26T14:30:24Z",
      "event_type": "vm_compliance_check"
    },
    {
      "href": "http://localhost:3000/api/vms/171/compliances/41",
      "id": "41",
      "resource_id": "171",
      "resource_type": "VmOrTemplate",
      "compliant": false,
      "timestamp": "2020-03-26T20:11:53Z",
      "updated_on": "2020-03-26T20:11:53Z",
      "event_type": "vm_compliance_check"
    }
  ],
  "links": {
    "self": "http://localhost:3000/api/vms/171/compliances?expand=resources&offset=0",
    "first": "http://localhost:3000/api/vms/171/compliances?expand=resources&offset=0",
    "last": "http://localhost:3000/api/vms/171/compliances?expand=resources&offset=0"
  }
}
```


Querying the virtual machine and its latest compliance result including the last compliance status

Request:

```
GET /api/vms/171?attributes=last_compliance,last_compliance_status
```

Response:


```
{
  "href": "http://localhost:3000/api/vms/171",
  "id": "171",
  "vendor": "vmware",
  "format": null,
  "version": null,
  "name": "aab-ldap",
  "description": null,
  "location": "aab-ldap/aab-ldap.vmx",
  "config_xml": null,
  "autostart": null,
  "host_id": 2,
  "last_sync_on": null,
  "created_on": "2020-03-10T18:53:36Z",
  "updated_on": "2020-03-26T01:58:16Z",
  "storage_id": "7",
  "guid": "30be2fa7-6b95-4a62-a163-d3107fc79d94",
  "ems_id": "2",
  "last_scan_on": "2020-03-18T21:13:32Z",
  "last_scan_attempt_on": "2020-03-18T21:12:25Z",
  "uid_ems": "4cda25de-c41c-4f68-8b3d-0cca0dc993c4",
  "retires_on": null,
  "retired": null,
  "boot_time": null,
  "tools_status": null,
  "standby_action": null,
  "power_state": "on",
  "state_changed_on": "2020-03-10T18:53:36Z",
  "previous_state": null,
  "connection_state": "connected",
  "last_perf_capture_on": "2020-03-26T01:58:16Z",
  "registered": null,
  "busy": null,
  "smart": null,
  "memory_reserve": null,
  "memory_reserve_expand": null,
  "memory_limit": null,
  "memory_shares": null,
  "memory_shares_level": null,
  "cpu_reserve": null,
  "cpu_reserve_expand": null,
  "cpu_limit": -1,
  "cpu_shares": 200,
  "cpu_shares_level": "normal",
  "cpu_affinity": null,
  "ems_created_on": null,
  "template": false,
  "evm_owner_id": null,
  "miq_group_id": "1",
  "linked_clone": true,
  "fault_tolerance": null,
```

```
  "type": "ManageIQ::Providers::Vmware::InfraManager::Vm",
  "ems_ref": "vm-825",
  "ems_cluster_id": "3",
  "retirement_warn": null,
  "retirement_last_warn": null,
  "vnc_port": null,
  "flavor_id": "960",
  "availability_zone_id": "23",
  "cloud": true,
  "retirement_state": null,
  "cloud_network_id": null,
  "cloud_subnet_id": null,
  "cloud_tenant_id": "2",
  "raw_power_state": "ACTIVE",
  "publicly_available": null,
  "orchestration_stack_id": null,
  "retirement_requester": null,
  "tenant_id": "1",
  "resource_group_id": null,
  "deprecated": null,
  "storage_profile_id": null,
  "cpu_hot_add_enabled": null,
  "cpu_hot_remove_enabled": false,
  "memory_hot_add_enabled": true,
  "memory_hot_add_limit": null,
  "memory_hot_add_increment": null,
  "hostname": null,
  "ems_ref_type": "VirtualMachine",
  "restart_needed": null,
  "last_compliance_status": false,
  "last_compliance": {
    "id": "41",
    "resource_id": "1000000132824",
    "resource_type": "VmOrTemplate",
    "compliant": false,
    "timestamp": "2020-03-26T20:11:53Z",
    "updated_on": "2020-03-26T20:11:53Z",
    "event_type": "vm_compliance_check"
  },
  "actions": [
    {
      "name": "edit",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "add_event",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "refresh",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "shutdown_guest",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "reboot_guest",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "start",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "stop",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "suspend",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
```

```
      "name": "shelve",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "shelve_offload",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "pause",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "request_console",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "reset",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "request_retire",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "delete",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "set_owner",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "set_ownership",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "set_miq_server",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "scan",
      "method": "post",
      "href": "http://localhost:3000/api/vms/171"
    },
    {
      "name": "delete",
      "method": "delete",
      "href": "http://localhost:3000/api/vms/171"
    }
  ]
}
```


Managed services APIs

This section includes the prerequisites to use Managed services APIs, Common Managed services APIs, sample Python script, and API reference.

- API reference
- Prerequisites
- Common APIs
- Sample Python script
- Content APIs
- Retrieve deployed APIs

API reference

You can use REST APIs to integrate Managed services with your application.

About the task

The terms in the swagger files for your reference:

- Stack is the API name for a deployed template instance.
- Stackstate is a separate table that keeps a history of all the detailed states the stack has been in. When you do a **`stack/<stack_id>/retrieve`**, it includes the latest state.
- Foreign key (fk) is the ID of that particular stackstate.

Procedure

1. Before you run API commands, generate access tokens. For more information about the steps, see Generating access tokens.
2. Copy the following Swagger files and save them in yaml format:
   - **`Managed services: cam-iaas`**

     The Managed services IaaS component provides support for defining connections to various clouds, templates that can be used to deploy stack instances.
   - **`Managed services: cam-service-composer-api`**

     The Managed services service composer.
   - **`Managed services: cam-tenant-api`**

     The Managed services tenant service provides a way to retrieve information about the tenant for the current user.
3. Managed services API needs tenantId, IBM Cloud Pak for AIOps team (ace_orgGuid), and IBM Cloud Pak for AIOps namespace (cloudOE_spaceGuid) for authorization. For the procedure to get these values, see Prerequisites for using Managed services APIs topic.
4. Load the yaml files onto any swagger browser.

Prerequisites for using Managed services APIs

This section includes prerequisite steps to use Managed services APIs.

The following table describes the variables that are used throughout the API calls.

Variable	Description
TSA_HOSTNAME	**`TSA_HOSTNAME`** is the **`HOST/PORT`** value in the output of the command: oc get routes cam -o=jsonpath='{.spec.host}'.
TSA_USER	**`TSA_USER`** is the IBM Cloud Pak for AIOps user you want to use. You can use either the default admin user or any user you have added/configured in the IBM Cloud Pak for AIOps.
TSA_USER_PASSWORD	**`TSA_USER_PASSWORD`** is the password for the **`TSA_USER`**.
MCM_CLUSTER_HOSTNAME	**`MCM_CLUSTER_HOSTNAME`** is the cluster hostname for IBM Cloud Pak for AIOps. It is the **`HOST/PORT`** value in the output of the command: **`$ oc -n ibm-common-services get route cp-console`**

1. Preparing to run Managed services APIs:
   1. Set the following environment variables:
      1. Run **`vi setenv`** to create a script.
      2. In **`setenv`**, paste the following environment variables:
         - **`export TSA_HOSTNAME=<Replace_with_your_TSA_HOSTNAME>`**
         - **`export TSA_USER=<Replace_with_your_CloudPak_for_MCM_User>`**
         - **`export TSA_USER_PASSWORD=<Replace_with_your_TSA_ADMIN_USER_PASSWORD>`**
         - **`export MCM_CLUSTER_HOSTNAME=<Replace_with_your_MCM_CLUSTER_HOSTNAME>`**
   2. Run **`chmod +x setenvAdd`** to execute permissions.
   3. Run **`. ./setenv`** script within the context of your current session.
2. To get an access token to your API calls and authenticate your credentials, see Get access token by using your username and password.
3. Most Managed services APIs require **`tenantId`** and IBM Cloud Pak for AIOps **`team (ace_orgGuid)`** as query parameters. Managed services APIs related to deployed template or service instances also require **`namespace (cloudOE_spaceGuid)`** as a query parameter.

The "Admin" user has access to all teams and namespaces.

To determine the Managed services **tenantId**, IBM Cloud Pak for AIOps **team**, and **namespace** issue the following API by using the access token from the previous API:

```
curl -k -X GET -H "Authorization: Bearer ${TSA_BEARER_TOKEN}"
https://${TSA_HOSTNAME}/cam/tenant/api/v1/tenants/getTenantOnPrem
```

- For tenantId, save the "id" value from the output:

  ```
  export TSA_TENANT_ID=<response.id>
  ```

- For IBM Cloud Pak for AIOps **team (ace_orgGuid)**, save the "teamId" from the namespace you would like to use:

  ```
  export MCM_TEAM=<response.namespaces[0].teamId>
  ```

- For **namespace (cloudOE_spaceGuid)**, save the "uid" from the namespace you would like to use:

  ```
  export NAMESPACE=<response.namespaces[0].uid>
  ```

  The following examples of getTenantOnPrem API are for "Admin" user and LDAP user:

  - When you call **https://${TSA_HOSTNAME}/cam/tenant/api/v1/tenants/getTenantOnPrem** with the admin account

    ```
    {
      "id": "1ed535a5-c895-4e5c-85fe-d8403a6c5331",  <-- tenantId
      "name": "ICP Account",
      "ownerId": "On_Prem",
      "wlpClientId": "XRw1JM5pQ0scGpwTNzN2gA3dTL1xDt07",
      "namespaces": [
       {
         "name": "default",
         "uid": "default",      <-- namespace (cloudOE_spaceGuid)
         "teamId": "all",         <-- MCM team (ace_orgGuid)
         "teamName": "all",
         "roles": [
          {
            "id": "crn:v1:icp:private:iam::::role:Administrator"
          }
                ]
              },
        {
          "name": "kube-public",
          "uid": "kube-public",
          "teamId": "all",
          "teamName": "all",
          "roles": [
          {
             "id": "crn:v1:icp:private:iam::::role:Administrator"
          }
                ]
          },
        {
          "name": "kube-system",
          "uid": "kube-system",
          "teamId": "all",
          "teamName": "all",
          "roles": [
            {
              "id": "crn:v1:icp:private:iam::::role:Administrator"
            }
              ]
         },
         {
           "name": "services",
           "uid": "services",
           "teamId": "all",
           "teamName": "all",
           "roles": [
             {
               "id": "crn:v1:icp:private:iam::::role:Administrator"
             }
                ]
           }
                ]
        }
    ```

  - For the "admin" user example, the following output must be saved:

    - **tenantId = response.id = 1ed535a5-c895-4e5c-85fe-d8403a6c5331**

```
export TSA_TENANT_ID=1ed535a5-c895-4e5c-85fe-d8403a6c5331
```

- IBM Cloud Pak for AIOps `team (ace_orgGuid) = response.namespaces[0].teamId = all`

  ```
  export MCM_TEAM=all
  ```

  Note: The admin does not belong to a team so use the value "all".

- IBM Cloud Pak for AIOps `namespace (cloudOE_spaceGuid)= response.namespaces[0].uid = default`

  ```
  export NAMESPACE=default
  ```

- When calling `https://${TSA_HOSTNAME}/cam/tenant/api/v1/tenants/getTenantOnPrem` with an LDAP user who is assigned to team "camusers" with access to the "default" namespace.

```
  {
    "id": "1ed535a5-c895-4e5c-85fe-d8403a6c5331",   <-- tenantId
    "name": "ICP Account",
    "ownerId": "On_Prem",
    "wlpClientId": "kSNhTWdjPWGVhqk8RdHcuNVl19uOCkrf",
    "namespaces": [
     {
       "name": "camdefault - default",
       "uid": "default",                              <-- namespace (cloudOE_spaceGuid)
       "teamId": "camdefault",                 <-- MCM team (ace_orgGuid)
       "teamName": "camdefault",
       "roles": [
        {
          "id": "crn:v1:icp:private:iam::::role:Editor"
        }
               ]
             }
          ]
  }
```

For the LDAP user example, the following output can be saved:

- `tenantId = response.id = 1ed535a5-c895-4e5c-85fe-d8403a6c5331`

  ```
  export TSA_TENANT_ID=1ed535a5-c895-4e5c-85fe-d8403a6c5331
  ```

- IBM Cloud Pak for AIOps `team (ace_orgGuid) = response.namespaces[0].teamId = camdefault`

  ```
  export MCM_TEAM=camdefault
  ```

- `namespace (cloudOE_spaceGuid)= response.namespaces[0].uid = default`

  ```
  export NAMESPACE=default
  ```

Common Managed services APIs

This section includes common Managed services APIs with examples.

- Provider and connection APIs
- Template and template instance APIs
- Terraform version APIs
- Terraform file APIs
- Resource specific APIs
- Service and service instance APIs
- Dynamic parameter discovery APIs
- Application APIs
- Shared parameter APIs
- SMTP connection APIS
- Usage specific APIs
- Example: Retrieve deployed templates list

Provider and connection APIs

This section includes all provider and connection related APIs with examples.

- Accessing APIs
- Retrieve the list of providers
- Create a cloud connection
- Retrieve the list of cloud connections
- Retrieve a specific cloud connection
- Update a specific cloud connection
- Delete a specific cloud connection

Accessing APIs

To get an access token to your API calls and authenticate your credentials, see Accessing APIs.

Retrieve the list of providers

Retrieve the list of providers.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/providers?tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Example response:

```
[
    {
        "configuration_parameters": [],
        "connection_parameters": [
            {
                "name": "ibmcloud_api_key",
                "type": "password",
                "description": "The API key for IBM Cloud",
                "required": "true",
                "secured": "true",
                "label": "IBM Cloud API Key",
                "value": ""
            },
            {
                "name": "iaas_classic_username",
                "type": "string",
                "description": "The User Name for IBM Cloud Classic Infrastructure",
                "required": "true",
                "secured": "false",
                "label": "IBM Cloud Classic Infrastructure User Name",
            },
            {
                "name": "iaas_classic_api_key",
                "type": "password",
                "description": "The API key for IBM Cloud Classic Infrastructure",
                "required": "true",
                "secured": "true",
                "label": "IBM Cloud Classic Infrastructure API Key",
                "value": ""
            }
        ],
        "name": "IBM",
        "description": "IBM Provider",
        "datatype_identifier": "IBM",
        "id": "5c588624df7d9e00187628c7",
        "label": "IBM"
    },
...
```

Create a cloud connection

POST to create a cloud connection.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create.json \
"https://${TSA_HOSTNAME}/cam/api/v1/cloudconnections?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

cc_create.json

```
{
  "name": "AWSTestConnection",
  "description": "My AWS Test Connection",
  "connection_parameters": [
    {
      "name": "aws_access_key_id",
      "type": "string",
      "description": "The API Access Key ID for AWS generated in the Identity and Access Management (IAM)
service",
      "required": "true",
      "secured": "true",
      "label": "Access Key ID",
      "default": null,
      "options": null,
      "value": "1234"
    },
    {
      "name": "aws_secret_key",
      "type": "password",
      "description": "The API Secret Access Key for AWS generated in the Identity and Access Management (IAM)
service",
      "required": "true",
      "secured": "true",
      "label": "Secret Access Key",
      "default": null,
      "options": null,
      "value": "5678"
    },
    {
      "name": "aws_default_region",
      "type": "string",
      "description": "Region to deploy the service on.",
      "required": "true",
      "secured": "false",
      "label": "AWS Region",
      "default": null,
      "options": [
        {
          "label": "US East (Northern Virginia)",
          "value": "us-east-1",
          "default": "true"
        },
        {
          "label": "US East (Ohio) Region",
          "value": "us-east-2"
        },
        {
          "label": "US West (Northern California)",
          "value": "us-west-1"
        },
        {
          "label": "US West (Oregon) Region",
          "value": "us-west-2"
        },
        {
          "label": "Asia Pacific (Seoul) Region",
          "value": "ap-northeast-2"
        },
        {
          "label": "Asia Pacific (Singapore) Region",
          "value": "ap-southeast-1"
        },
        {
          "label": "Asia Pacific (Sydney) Region",
          "value": "ap-southeast-2"
        },
        {
          "label": "Asia Pacific (Tokyo) Region",
          "value": "ap-northeast-1"
        },
        {
          "label": "EU (Frankfurt) Region",
          "value": "eu-central-1"
        },
        {
          "label": "EU (Ireland) Region",
          "value": "eu-west-1"
        },
        {
          "label": "EU (London) Region",
          "value": "eu-west-2"
```

```
        },
        {
          "label": "AWS GovCloud (US)",
          "value": "us-gov-west-1"
        }
      ],
      "value": "us-east-1"
    }
  ],
  "configuration_parameters": [
  ],
  "providerId": "5bb76eb78e04ec001c19466f",
  "additional_Parameters": [
    {
      "plugin": "plugin-terraform",
      "supported_api_types": [
        "stack",
        "iaas"
      ],
      "supported_template_types": [
        {
          "template_type": "Terraform",
          "template_formats": [
            "JSON",
            "HCL"
          ]
        }
      ],
      "configuration_parameters": [
      ]
    }
  ]
}
```

- Provider Id - Using the Retrieve the list of providers GET API call, retrieve the Provider id. In this example, AWS provider is used.
- `cloudOE_spaceGuid` query parameter governs the namespace to be used when a cloud connection is created. In this example, the cloud connection gets created in the namespace that `$NAMESPACE` resolves to. Now, if you want to create the cloud connection to be globally available and not tied to any specific namespace, then omit the cloudOE_spaceGuid query parameter.

GET the list of cloud connections

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/cloudconnections?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example output:

```
[
    {
        "name": "MyVMware",
        "description": "My VMware vSphere Connection",
        "connection_parameters": [
            {
                "name": "vsphere_user",
                "value": "username@vsphere.local",
                "secured": "false"
            },
            {
                "name": "vsphere_password",
                "value": "*****",
                "secured": "true"
            },
            {
                "name": "vsphere_server",
                "value": "1.2.3.4",
                "secured": "false"
            }
        ],
        "validate": true,
        "message": "",
        "id": "59f3b11bd1b007002263c5f3",
        "providerId": "59f38667d1b007002263c578",
        "additional_Parameters": {
            "0": {
                "plugin": "plugin-terraform"
            }
        },
        "tenantId": "5c0e0a5e-d431-468c-9a11-a3df3978e424"
```

```
    }
]
```

The **cloudOE_spaceGuid** query parameter is optional and the search results display all cloud connections that **$NAMESPACE** resolves to.

GET a specific cloud connection (filter)

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/cloudconnections?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&filter=%7B%22where%22%3A%20%7B%22name%22%3A%20%22vmware.oct
ravis%22%20%7D%7D"
```

Example output:

```
[
  {
    "name": "vmware.octravis",
    "description": "vmware.octravis",
    "connection_parameters": [
      {
        "name": "vsphere_user",
        "value": "octravis@vsphere.local",
        "secured": "false"
      },
      {
        "name": "vsphere_password",
        "value": "*****",
        "secured": "true"
      },
      {
        "name": "vsphere_server",
        "value": "1.2.3.123",
        "secured": "false"
      }
    ],
    "configuration_parameters": [
      {
        "name": "",
        "value": ""
      }
    ],
    "validate": true,
    "message": "",
    "id": "5a4d094a47ff9c0022728759",
    "providerId": "5a4cf14b47ff9c0022728676",
    "additional_Parameters": [
      {
        "plugin": "plugin-terraform"
      }
    ],
    "tenantId": "15e7d67e-9f73-43cd-9759-b39ebeea85e8"
  }
]
```

Update a specific cloud connection

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_update.json \
"https://${TSA_HOSTNAME}/cam/api/v1/cloudconnections/${CAM_CLOUD_CONNECTION_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example request body:

```
{
  "name": "Updated connection name",
  "description": "Updated connection description"
}
```

Delete a specific cloud connection

```
curl -k -X DELETE \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
```

```
"https://${TSA_HOSTNAME}/cam/api/v1/cloudconnections/${CAM_CLOUD_CONNECTION_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Template and template instance APIs

This section includes template and template instance APIs with examples.

- Create a template from a file
- Create a template from a source
- Duplicate a template
- Retrieve the list of templates
- Retrieve a specific template using filter
- Retrieve a specific template by id
- Create new template version for an existing template
- Deploy a template
- Retrieve the list of deployed templates
- Retrieve details for a template instance
- Modify a template instance
- Apply the changes to a template instance
- Destroy a template instance
- Delete a template instance (stack) after destroying it
- Retrieve stack information
- Download the Terraform state (tfstate) for a template instance
- Download the Terraform state (tfstate) for a specific stack state
- Upload the Terraform state (tfstate) for a template instance
- Download the execution plan for a template instance
- Retrieve the execution plan in json format for a template instance

Create a template from a file

```
curl -k -X POST \
-H "Authorization: ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @template_create.json \
"https://${TSA_HOSTNAME}/cam/api/v1/templates?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

```
template_create.json:
{
    "name": "mytemplate",
    "description": "here is the description",
    "type": "userCreated",
    "manifest": {
        "template_type": "Terraform",
        "template": {
            "templateData": "variable \"region\" {\n   default = \"us-east-1\"\n}\n",
            "templateOutput": "",
            "templateVariables": {
                "template_input_params": [
                    {
                        "name": "region",
                        "description": "variable description",
                        "default": "us-east-1",
                        "type": "string",
                        "required": true,
                        "secured": false,
                        "label": "region",
                        "hidden": false,
                        "immutable": false
                    }
                ]
            }
        },
        "template_provider": "IBM"
    },
    "metadata": {
        "displayName": "the template display name"
    }
}
```

The **cloudOE_spaceGuid** query parameter governs the namespace to be used when a template is created. In this example, the template gets created in the namespace that **$NAMESPACE** resolves to. Now, if you want to create the template to be globally available and not tied to any specific namespace, then omit the **cloudOE_spaceGuid** query parameter.

Example output:

```
{
    "description": "here is the description",
    "name": "mytemplate",
    "type": "userCreated",
    "manifest": {
        "template_type": "Terraform",
        "template_provider": "IBM",
        "template": {
            "templateData": "variable \"region\" {\n   default = \"us-east-1\"\n}\n",
            "templateVariables": {
                "template_input_params": [
                    {
                        "name": "region",
                        "description": "variable description",
                        "default": "us-east-1",
                        "type": "string",
                        "required": true,
                        "secured": false,
                        "label": "region",
                        "hidden": false,
                        "immutable": false
                    }
                ]
            }
        }
    },
    "metadata": {
        "displayName": "the template display name"
    },
    "created_at": "2018-01-09T05:10:48.523Z",
    "id": "5a544ed88bec4000200dc328",
    "tenantId": "cd3ed377-8167-4078-b486-309b1f237928",
    "default_template_version": "5a544ed88bec4000200dc329",
    "template_version": {
        "id": "5a544ed88bec4000200dc329",
        "name": "v1.0.0",
        "enabled": true
    },
    "template_versions": [
        {
            "id": "5a544ed88bec4000200dc329",
            "name": "v1.0.0",
            "enabled": true
        }
    ]
}
```

Create a template from a source

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @template_input.json \
"https://${TSA_HOSTNAME}/cam/api/v1/templates/createFromSource?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

template_input.json:

```
{
    "template_type": "Terraform",
    "template_format": "HCL",
    "template_source": {
        "github": {
            "url": "https://github.com/org/provider-terraform-base",
            "token": "access token for github repository",
            "dir": "subdir/testTemplate",
            "ref": "master"
        }
    }
}
```

The **cloudOE_spaceGuid** query parameter governs the namespace to be used when a template is created. In this example, the template gets created in the namespace that **$NAMESPACE** resolves to. Now, if you want to create the template to be globally available and not tied to any specific namespace, then omit the **cloudOE_spaceGuid** query parameter.

Example output:

```
{
    "description": "Template contents are hosted in a git repository.",
    "name": "New template hosted in git 129c26100b5b9fe7",
    "type": "userCreated",
    "manifest": {
        "template_type": "Terraform",
        "template_source": {
            "github": {
                "url": "https://github.com/org/provider-terraform-base",
                "token": "1234567890",
                "dir": "subdir/testTemplate1",
                "ref": "master"
            },
            "displayUrl": "https://github.com/org/provider-terraform-base/tree/master/subdir/testTemplate1",
            "displayUrlReplaceRef": "https://github.com/org/provider-terraform-
base/tree/REPLACE_REF/subdir/testTemplate1"
        },
        "template": {
            "templateVariables": {}
        }
    },
    "metadata": {
        "displayName": "",
        "longDescription": ""
    },
    "created_at": "2018-01-09T05:24:30.288Z",
    "id": "5a54520e8bec4000200dc32a",
    "tenantId": "cd3ed377-8167-4078-b486-309b1f237928",
    "default_template_version": "5a54520e8bec4000200dc32b",
    "template_version": {
        "id": "5a54520e8bec4000200dc32b",
        "ref": "master",
        "name": "master",
        "enabled": true,
        "displayUrl": "https://github.com/org/provider-terraform-base/tree/master/subdir/testTemplate1"
    },
    "template_versions": [
        {
            "id": "5a54520e8bec4000200dc32b",
            "ref": "master",
            "name": "master",
            "enabled": true,
            "displayUrl": "https://github.com/org/provider-terraform-base/tree/master/subdir/testTemplate1"
        }
    ]
}
```

Example data to import a template whenever you use a compressed file:

```
{
"template_type": "Terraform",
"template_source": {
  "archive": {
    "url": "https://myserver.myorg.com/templates/template1.zip",
  }
}
}
```

The **artifactory_api_key** is an optional parameter that allows you to specify an API key to an artifactory location. If the file is hosted in artifactory and an API key is needed, then use the **artifactory_api_key**. For example:

```
{
"template_type": "Terraform",
"template_source": {
  "archive": {
    "url": "https://myserver.myorg.com/templates/template1.zip",
    "artifactory_api_key": "YOURKEYHERE"
  }
}
}
```

If you get an error during the API call, ensure that your archive is in "zip" format and your file is downloadable through HTTP/HTTPS.

Duplicate a template

This API duplicates the template along with all versions.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @duplicate_template.json \
"https://${TSA_HOSTNAME}/cam/api/v1/templates/${CAM_TEMPLATE_ID}/duplicate?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

The **cloudOE_spaceGuid** query parameter governs the namespace to be used when the duplicate template is created. In this example, the template gets created in the namespace that **$NAMESPACE** resolves to. Now, if you want to create the template to be globally available and not tied to any specific namespace, then omit the **cloudOE_spaceGuid** query parameter.

Example request body:

```
{
 "name": "mynewname"
}
```

Retrieve the list of templates

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/templates?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

The **cloudOE_spaceGuid** query parameter is optional and the search results display all templates that are created for the namespace **$NAMESPACE**. The result also includes all the templates that are globally available.

Example output:

```
[
    {
        "description": "Kubernetes - An orchestration system for containerized applications, e.g., Nginx.",
        "name": "Kubernetes Cluster with Nginx Deployment",
        "type": "prebuilt",
        "manifest": {
            "template_type": "Terraform",
            "template_format": "HCL",
            "template_provider": "VMware vSphere"
        },
        "metadata": {
            "displayName": "Kubernetes Cluster with Nginx Deployment",
            "longDescription": "Deploys a Kubernetes Cluster in VMware, where Kubernetes is an open-source
system for automating deployment, scaling, and management of containerized applications. This deployment also
deploys the NGINX sample application to validate the deployment is operational, where NGINX is an open source
web server and load balancer for high-traffic sites.",
            "bullets": [
                {
                    "title": "Clouds",
                    "description": "VMware"
.
.
        "created_at": "2017-10-27T19:17:59.874Z",
        "id": "59f38667d1b007002263c581",
        "tenantId": "DEFAULT",
        "default_template_version": "59f38667d1b007002263c582"
    },
```

Retrieve a specific template using filter

This API returns the template metadata along with all the template versions.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/templates?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&filter=%7B%22where%22%3A%20%7B%22name%22%3A%20%22SingleVirt
ualMachine%22%20%7D%7D"
```

Example output:

```
[
    {
```

```
        "description": "Create Virtual Machine with single vnic",
        "name": "SingleVirtualMachine",
        "type": "prebuilt",
        "manifest": {
            "template_type": "Terraform",
            "template_format": "HCL",
            "template_provider": "VMware vSphere"
        },
        "metadata": {
            "displayName": "SingleVirtualMachine",
            "longDescription": "Create Virtual Machine with single vnic on a network label by cloning"
        },
        "created_at": "2017-11-03T15:19:30.566Z",
        "id": "59fc89024722a1001ff2974c",
        "tenantId": "DEFAULT",
        "default_template_version": "59fc89024722a1001ff2974d"
    }
]
```

Retrieve a specific template by id

This API returns the template metadata along with all the template versions.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/templates/${CAM_TEMPLATE_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Create new template version for an existing template

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @template_version.json \
"https://${TSA_HOSTNAME}/cam/api/v1/templateVersions?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Examples of input template_version.json:

- Create a new template version from git:

  ```
  {
   "name": "v2.0",
   "ref": "master",
   "templateId": "parent template id"
  }
  ```

- Create a template version from file:

  ```
  {
   "name": "v2.0",
   "templateId": "parent template id",
   "templateData": "variable \"region\" {\n   default = \"us-east-1\"\n}\n",
   "templateVariables": {
         "template_input_params": [
               {
                   "name": "region",
                   "description": "variable description",
                   "default": "us-east-1",
                   "type": "string",
                    "required": true,
                    "secured": false,
                    "label": "region",
                    "hidden": false,
                    "immutable": false
               }
          ]
    }
  }
  ```

Deploy a template

Get the template versions using the Retrieve a specific template by id if you want to specify a particular template version while deploying the template.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @template_input.json \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example input:

```
template_input.json:
{
  "name": "vSphere95",
  "cloud_connection_ids": [
    "5a00ab148af3f3002010ab45"
  ],
  "parameters": [
    {
      "name": "datacenter",
      "value": "Rack-8283_1000167"
    },
    {
      ....
    },
  ]
  "templateId": "5a00a2448af3f3002010aae1",
  "templateVersionId": "5a00a2448af3f3002010aae2",
  "catalogName": "SingleVirtualMachine",
  "catalogType": "userCreated",
  "forceCreate": "true"
  }
```

Note: **templateVersionId** is an optional parameter, if it is not provided, the default provided template version is used for deployment.

When the template to be deployed has input_datatypes, then use the following information:

Pass the input dataobjects that you want to use as your input to a stack by using the name and Data Type. It uniquely identifies the dataobject instance you want to use. For example:

```
"input_dataobjects": [
    {
      "name": "DefaultNoBastionHostRequired",
      "datatype": "bastionhost"
    }
  ],
```

Where:

- **bastionhost** is the Data Type name. It matches the datatype in the **input_datatypes** of the template you are deploying.
- **name** is the name of the Data Object instance you want to use when you deploy this instance.

For each input_datatype, the template requires an input_dataobject.

In this example, the template requires an input_datatype of "bastionhost". When you deploy a stack from this template, provide the dataobject to use that implements the "bastionhost" datatype:

```
"input_dataobjects": [
    {
      "name": "DefaultNoBastionHostRequired",
      "datatype": "bastionhost"
    }
  ],
```

Example Response:

```
{
  "name": "vSphere95",
  "created_at": "2017-11-06T20:18:11.697Z",
  "templateId": "5a00a2448af3f3002010aae1",
  "namespaceId": "default",
  "templateName": "SingleVirtualMachine",
  "templateCategory": "prebuilt",
  "templateProvider": "VMware vSphere",
  "template_type": "Terraform",
  "template_format": "HCL",
  "applied_state_id": "5a00c3838af3f3002010ab53",
  "latest_state_id": "5a00c3838af3f3002010ab53",
  "cloud_connection_ids": [
```

```
    "5a00ab148af3f3002010ab45"
  ],
  "id": "5a00c3838af3f3002010ab53",
  "tenantId": "df87bf8a-2ea9-478d-9b37-2b36da4af6ed",
  "status": "IN_PROGRESS",
  "message": "",
  "action": "CREATE",
  "stack_id": "9cb55c20-c32f-11e7-8d7f-d1b85b52d79c",
  "applied_status": "IN_PROGRESS",
  "applied_action": "CREATE",
  "data": {
    "stack_id": "9cb55c20-c32f-11e7-8d7f-d1b85b52d79c",
    "stack_job_id": "5a00c3838af3f3002010ab53",
    "stack_job_is_latest": true,
    "action": "CREATE",
    "status": "IN_PROGRESS",
    "tenantId": "df87bf8a-2ea9-478d-9b37-2b36da4af6ed"
  }
}
```

Create a stack (advanced scenario)

Use this procedure if you want to do a plan on the stack before you deploy:

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @template_input.json \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example input:

```
{
  "name": "vSphere95",
  "cloud_connection_ids": [
    "5a00ab148af3f3002010ab45"
  ],
  "parameters": [
    {
      "name": "datacenter",
      "value": "Rack-8283_1000167"
    },
    {
      ....
    },
  ]
  "templateId": "5a00a2448af3f3002010aae1",
  "templateVersionId": "5a00a2448af3f3002010aae2",
  "catalogName": "SingleVirtualMachine",
  "catalogType": "userCreated",
  "forcePlan": "true"
  }
```

Retrieve the list of deployed templates

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example output:

```
[
  {
        "name": "My WAS Liberty",
        "created_at": "2017-10-29T16:03:20.985Z",
        "templateId": "59f38692d1b007002263c5e3",
        "namespaceId": "default",
        "templateName": "IBM WebSphere Application Server Liberty V17 on a single virtual machine - VMware
vSphere",
        "templateCategory": "chefcontent",
        "templateProvider": "VMware vSphere",
        "template_type": "Terraform",
        "template_format": "HCL",
        "applied_state_id": "59f5fbc9d1b007002263c5fa",
        "latest_state_id": "59f5fbc9d1b007002263c5fa",
        "cloud_connection_ids": [
```

```
            "59f3b11bd1b007002263c5f3"
        ],
        "id": "59f5fbc8d1b007002263c5f9",
        "tenantId": "5c0e0a5e-d431-468c-9a11-a3df3978e424",
        "status": "SUCCESS",
        "message": "",
        "action": "CREATE",
        "stack_id": "afa51b40-bcc2-11e7-bbfe-b56e007b5b00",
        "applied_status": "SUCCESS",
        "applied_action": "CREATE",
        "applied_output_namespace": "content_template_output"
    }
]
```

The **cloudOE_spaceGuid** query parameter is optional and the search results display all template instances that **$NAMESPACE** resolves to.

Retrieve details for a template instance

Example:

```
export STACK_ID=5a00c3838af3f3002010ab53
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${STACK_ID}/retrieve?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example response:

```
{
  "name": "vSphere95",
  "created_at": "2017-11-06T20:18:11.697Z",
  "templateId": "5a00a2448af3f3002010aae1",
  "namespaceId": "default",
  "templateName": "SingleVirtualMachine",
  "templateCategory": "prebuilt",
  "templateProvider": "VMware vSphere",
  "template_type": "Terraform",
  "template_format": "HCL",
  "applied_state_id": "5a00c3838af3f3002010ab54",
  "latest_state_id": "5a00c3838af3f3002010ab54",
  "cloud_connection_ids": [
    "5a00ab148af3f3002010ab45"
  ],
  "id": "5a00c3838af3f3002010ab53",
  "tenantId": "df87bf8a-2ea9-478d-9b37-2b36da4af6ed",
  "status": "SUCCESS",
  "message": "",
  "action": "CREATE",
  "stack_id": "9cb55c20-c32f-11e7-8d7f-d1b85b52d79c",
  "applied_status": "SUCCESS",
  "applied_action": "CREATE",
  "data": {
    "stack_id": "9cb55c20-c32f-11e7-8d7f-d1b85b52d79c",
    "stack_job_id": "5a00c3834e13b5001f4aecd1",
    "stack_job_is_latest": true,
    "status": "SUCCESS",
    "action": "CREATE",
    "message": "\nInitializing provider plugins...\nTerraform has been successfully initialized!\n\nYou may
now begin working with Terraform. Try running \"terraform plan\" to see\nany changes that are required for
your infrastructure. All Terraform commands\nshould now work.\n\nIf you ever set or change modules or backend
configuration for Terraform,\nrerun this command to reinitialize your working directory. If you forget,
other\ncommands will detect it and remind you to do so if necessary.\ndata.vsphere_datacenter.datacenter:
Refreshing state...\nvsphere_virtual_machine.vm_1: Creating...\n  cluster:
```

Modify a template instance (stack)

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @plan_input.json \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${STACK_ID}/plan?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

```
plan_input.json:
{
  "templateVersionId": "59fc89024722a1001ff2974d",
  "parameters": [
```

```
    {
      "name": "datacenter",
      "value": "Rack-8283_1000167"
    },
    {
      ...
    },
  ]
 }
```

For the plan request body, the parameters are optional and the execution plan gets created by using the parameter values from the initial deploy.

Example response:


```
{
    "name":"vSphere95",
    "created_at":"2017-12-13T14:48:22.937Z",
    "templateId":"5a2b00d7b17b800020af2e7c",
    "namespaceId":"default",
    "templateName":"SingleVirtualMachine",
    "templateCategory":"prebuilt",
    "templateProvider":"VMware vSphere",
    "template_type":"Terraform",
    "applied_state_id":"5a313db7b17b800020af300a",
    "latest_state_id":"5a314639b17b800020af3023",
    "cloud_connection_ids":[
        "5a2b0513b17b800020af2f2b"
    ],
    "id":"5a313db6b17b800020af3009",
    "tenantId":"a8fbbe67-1480-49c0-be60-0084b31a0942",
    "status":"IN_PROGRESS",
    "message":"",
    "action":"PLAN",
    "stack_id":"ab100110-e014-11e7-85f3-c3bc8b9fdda1",
    "applied_status":"SUCCESS",
    "applied_action":"CREATE",
    "data":{
        "stack_id":"ab100110-e014-11e7-85f3-c3bc8b9fdda1",
        "stack_job_id":"5a3146396e53230020466164",
        "stack_job_is_latest":true,
        "action":"PLAN",
        "status":"IN_PROGRESS",
        "tenantId":"a8fbbe67-1480-49c0-be60-0084b31a0942"
    }
}
```


A successful plan has two states, namely SUCCESS AND SUCCESS_WITH_CHANGES:

- The SUCCESS indicates that the plan is successful and there will not be any changes to your environment even if you apply the plan.
- The SUCCESS_WITH_CHANGES indicates that the plan is successful, but when you apply it, there will be changes to your environment. Go through the logs to understand more about the changes to the environment. For example, a template is defined to create a virtual machine. You make modifications to the template such that it creates a second virtual machine. The resultant SUCCESS_WITH_CHANGES indicates that if you apply the plan, the second virtual machine will get created.

Apply the changes to a template instance (stack)

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${STACK_ID}/apply?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example output:


```
{
    "name":"vSphere95",
    "created_at":"2017-12-13T14:48:22.937Z",
    "templateId":"5a2b00d7b17b800020af2e7c",
    "namespaceId":"default",
    "templateName":"SingleVirtualMachine",
    "templateCategory":"prebuilt",
    "templateProvider":"VMware vSphere",
    "template_type":"Terraform",
    "applied_state_id":"5a314892b17b800020af3042",
    "latest_state_id":"5a314892b17b800020af3042",
    "cloud_connection_ids":[
        "5a2b0513b17b800020af2f2b"
    ],
    "id":"5a313db6b17b800020af3009",
```

```
    "tenantId":"a8fbbe67-1480-49c0-be60-0084b31a0942",
    "status":"IN_PROGRESS",
    "message":"",
    "action":"APPLY",
    "stack_id":"ab100110-e014-11e7-85f3-c3bc8b9fdda1",
    "applied_status":"IN_PROGRESS",
    "applied_action":"APPLY",
    "data":{
        "stack_id":"ab100110-e014-11e7-85f3-c3bc8b9fdda1",
        "stack_job_id":"5a3148926e53230020466174",
        "stack_job_is_latest":true,
        "action":"APPLY",
        "status":"IN_PROGRESS",
        "tenantId":"a8fbbe67-1480-49c0-be60-0084b31a0942"
    }
}
```

Destroy a template instance (stack)

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${STACK_ID}/delete?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example response:

```
  {
  "name": "vSphere95",
  "created_at": "2017-11-06T20:18:11.697Z",
  "templateId": "5a00a2448af3f3002010aae1",
  "namespaceId": "default",
  "templateName": "SingleVirtualMachine",
  "templateCategory": "prebuilt",
  "templateProvider": "VMware vSphere",
  "template_type": "Terraform",
  "template_format": "HCL",
  "applied_state_id": "5a00c4a08af3f3002010ab57",
  "latest_state_id": "5a00c4a08af3f3002010ab57",
  "cloud_connection_ids": [
    "5a00ab148af3f3002010ab45"
  ],
  "id": "5a00c3838af3f3002010ab53",
  "tenantId": "df87bf8a-2ea9-478d-9b37-2b36da4af6ed",
  "status": "SUCCESS",
  "message": "",
  "action": "DELETE",
  "stack_id": "9cb55c20-c32f-11e7-8d7f-d1b85b52d79c",
  "applied_status": "SUCCESS",
  "applied_action": "DELETE",
  "data": {
    "stack_id": "9cb55c20-c32f-11e7-8d7f-d1b85b52d79c",
    "stack_job_id": "5a00c4a04e13b5001f4aecd2",
    "stack_job_is_latest": true,
    "status": "SUCCESS",
    "action": "DELETE",
    "message": "\nInitializing provider plugins...\nTerraform has been successfully initialized!\n\nYou may
now begin working with Terraform.
```

Delete a template instance (stack) after destroying it

```
curl -k -X DELETE \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${STACK_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example response: `{"count":1}`

Retrieve stack information

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/{STACK_ID}/retrieve?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

This retrieves information about the stack including its parameters. However, if any of those parameters have **secured:true**, then the values for those parameters are hidden, that is, replaced with asterisks by default.

To return secured values back in plain text, run the POST with **showSecured=true** parameter.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/{STACK_ID}/retrieve?
showSecured=true&tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Download the Terraform state (tfstate) for a template instance

This API downloads the Terraform state (tfstate) of a template instance.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-o terraform.tfstate \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${CAM_STACK_ID}/tfstate?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Note: The **-o** file option is optional. It writes the output to a file instead of stdout.

Download the Terraform state (tfstate) for a specific stack state

This API downloads the Terraform state (tfstate) for a specific stack state.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-o terraform.tfstate \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${CAM_STACK_ID}/stackstates/${CAM_STACK_STATE_ID}/tfstate?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Upload the Terraform state (tfstate) for a template instance

This API uploads the Terraform state (tfstate) to a template instance.

You must have Administrator, cluster administrator, or editor role to run this API.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-F 'tfstate=@your_tfstate_file' \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${CAM_STACK_ID}/tfstate?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Download the execution plan for a template instance

This API downloads an execution plan for a template instance.

You must have Administrator, cluster administrator, or editor role to run this API.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-o terraform_execution_plan \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${CAM_STACK_ID}/executionPlan?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Note: The execution plan file is base64 encoded. You must decode the file content before using it.

Retrieve the execution plan in json format for a template instance

This API retrieves the execution plan in json format for a template instance. The API works only for template instances that use Terraform engine 0.12.x or newer.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${CAM_STACK_ID}/executionPlan/show?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Terraform version APIs

This section includes Terraform version APIs.

- Retrieve Terraform versions
- Create a Terraform version
- Update a Terraform version
- Delete a Terraform version
- Set Terraform version as Default
- Enable/disable a Terraform version

Retrieve Terraform versions

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/providerengineversions?tenantId=${TSA_TENANT_ID}"
```

Create a Terraform version

You must have Administrator or cluster administrator role to run this API.

```
curl -k -X POST -H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
  -F "template_type=Terraform" \
  -F "version=0.15.5" \
  -F "default=false" \
  -F "enabled=true" \
  -F "zipfile=@terraform_0.15.5.zip" \
  "https://${TSA_HOSTNAME}/cam/api/v1/providerengineversions?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Update a Terraform version

You must have Administrator or cluster administrator role to run this API.

```
curl -k -X PUT \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @update_terraform_version.json \
"https://${TSA_HOSTNAME}/cam/api/v1/providerengineversions/${TERRAFORM_VERSION_ID}?tenantId=${TSA_TENANT_ID}"
```

Example of the request body:

```
{
     version: '0.12.15',
     default: true
}
```

Delete a Terraform version

You must have Administrator or cluster administrator role to run this API.

```
curl -k -X DELETE \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/providerengineversions/${TERRAFORM_VERSION_ID}?tenantId=${TSA_TENANT_ID}"
```

Set Terraform version as default

You must have Administrator or cluster administrator role to run this API.

```
curl -k -X PUT \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @default_terraform_version.json \
"https://${TSA_HOSTNAME}/cam/api/v1/providerengineversions/${TERRAFORM_VERSION_ID}?tenantId=${TSA_TENANT_ID}"
```

Example of the request body:

```
{
     default: true
}
```

Enable/disable a Terraform version

You must have Administrator or cluster administrator role to run this API.

```
curl -k -X PUT \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @enable_terraform_version.json \
"https://${TSA_HOSTNAME}/cam/api/v1/providerengineversions/${TERRAFORM_VERSION_ID}?tenantId=${TSA_TENANT_ID}"
```

Example of the request body to enable the Terraform version:

```
{
    enabled: true
}
```

Example of the request body to disable the Terraform version:

```
{
    enabled: false
}
```

Terraform File APIs

This section includes Terraform file APIs. You can upload terraform files (plug-ins or other custom files) to provider terraform pods by using these APIs.

- Retrieve Terraform files
- Create a Terraform file
- Delete a Terraform file

Retrieve Terraform files

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/providerenginefiles?tenantId=${TSA_TENANT_ID}"
```

Create a Terraform file

You must have Administrator or cluster administrator role to run this API.

```
curl -k -X POST -H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
  -F "template_type=Terraform" \
  -F "type=plugin" \
  -F "zipfile=@customer_plugins.zip" \
  "https://${TSA_HOSTNAME}/cam/api/v1/providerenginefiles?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

In the above example, the user uploads a new terraform plug-in. The archive can contain multiple plug-ins and are copied to /home/terraform/.terraform.d/plugins on each provider terraform pod while restarted.

```
curl -k -X POST -H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
  -F "template_type=Terraform" \
  -F "type=file" \
  -F "zipfile=@customer_files.zip" \
  "https://${TSA_HOSTNAME}/cam/api/v1/providerenginefiles?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

In the above example, the user uploads a new terraform file (that is, certificate). The archive can contain multiple files and are copied to /home/terraform/.terraform.d/cam on each provider terraform pod when they are restarted.

Delete a Terraform file

You must have Administrator or cluster administrator role to run this API.

```
curl -k -X DELETE \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/providerenginefiles/${TERRAFORM_FILE_ID}?tenantId=${TSA_TENANT_ID}"
```

Resource-specific APIs

This section includes resource specific APIs with examples.

- Restart OS
- Import VM by IDs
- Import VM by Name
- Create snapshots
- Revert to snapshot
- Delete a snapshot
- Revert to current snapshot
- Get snapshots
- Taint a resource from a stack
- Untaint a resource from a stack
- Retrieve and change power states for resources
- Start a virtual machine
- Stop a virtual machine
- Reset virtual machine
- Get the status of a stopping or starting a virtual machine
- Retrieve all IaaS resource settings
- Retrieve a specific IaaS resource settings
- Create IaaS resource settings
- Update IaaS resource settings
- Delete IaaS resource settings

Note: There are three possible statuses for a resource task:

- IN_PROGRESS
- SUCCESS
- FAILED

Restart OS

The API call restarts the OS. It is available only for VMware vSphere virtual machine actions. The option is enabled only when the virtual machine powerstate is ON.

Note: The virtual machine**`resource.id`** can be retrieved from the update the state of the template instance API.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{"task_name":"restart"}' \
"https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

POST method.

```
Body:{ "task_name: "restart" }
```

Example response:

```
{
  "cloud_resource_id": "421ac557-7793-fdd5-7b83-c3525571abfe",
  "task_name": "restart",
  "status": "IN_PROGRESS",
  "created": "2017-11-13T16:47:58.298Z",
  "updated": "2017-11-13T16:47:58.298Z",
  "stack_id": "7d55d130-c239-11e7-b95f-c5cc4d245984",
  "resource_id": "59ff272d4ba47e0020cb6583",
  "resource_name": "vsphere_virtual_machine.vm_1",
  "id": "5a09ccbe9fe5890020316467"
}
```

Import VM by IDs

Imports an existing virtual machine by using the ID. You can view stack instance in the template deployed instances page.

```
https://${TSA_HOSTNAME}/cam/api/v1/stacks/import?tenantId=${TSA_TENANT_ID}&cloudOE_spaceGuid=${NAMESPACE}
```

POST method

For example, import a virtual machine's resources that are available in stack "fit-vm17-76":

```
Body
{
  "name": "fit-vm17-76",
  "cloudConnectionId": "5a8983938f22883ee775"
  "templateId": "5b39493e3984938943eb8b6",
  "templateVersionId: "5b39493e3984938943eb8c5",
  "idFromProvider": "/Rack-8283_1000167/vm/shared/fit-vm17-76"
}
```

Where,

- `name` is the name of the stack.
- `cloudConnectionId` is the ID of the connection where the virtual machine can be found. Use the credentials used in that cloud to find the virtual machine.
- `templateId` is to associate a template to a stack instance server.
- `templateVersionId` is the version of the template. It is an optional parameter.
- `idFromProvider` is the id of the virtual machine. The format differs for PowerVC and OpenStack.
- `resourceName` is an optional parameter. If resourceName is set, then you can import from providers other than vSphere and OpenStack. If you do not use this parameter, then the default values are as follows:
  - openstack_compute_instance_v2.vm_1 for OpenStack instances
  - vsphere_virtual_machine.vm_1 for VMWare vSphere instances

Import VM by Name

Imports an existing virtual machine by using the name. You can view the stack instance in the template deployed instances page.

```
https://${TSA_HOSTNAME}/cam/api/v1/stacks/import?tenantId=${TSA_TENANT_ID}&cloudOE_spaceGuid=${NAMESPACE}
```

POST method

For example, import a virtual machine resource from stack "fit-vm17-76":

```
Body
{
  "name": "fit-vm17-76",
  "cloudConnectionName": "fit3",
  "templateName": "Import existing VM with a single disk - VMware vSphere",
  "templateVersionName": "1.0",
  "idFromProvider": "/Rack-8283_1000167/vm/Shared/fit-vm17-76"
}
```

Where,

- `name` is the name of the stack.
- `CloudConnectionName` is the connection name where the virtual machine can be found. Use the credentials used in that cloud to find the virtual machine.
- `templateName` is to associate a template to a stack instance server. It can be VMware vSphere or PowerVC/OpenStack.
- `templateVersionName` is the version of the template. It is an optional parameter.
- `idFromProvider` is the id of the virtual machine. The format differs for PowerVC and OpenStack.
- `resourceName` is an optional parameter. If resourceName is set, then you can import from providers other than vSphere and OpenStack. If you do not use this parameter, then the default values are as follows:
  - openstack_compute_instance_v2.vm_1 for OpenStack instances
  - vsphere_virtual_machine.vm_1 for VMWare vSphere instances

Create snapshot

This API creates a virtual machine snapshot for a VMware instance.

Note: The virtual machine `resource.id` can be retrieved from the update the state of the template instance API.

Sample URL:

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{"task_name":"snapshot", "parameters": [ \
   { \
       "name": "snapshot_name", \
       "value": ${SNAPSHOT_NAME} \
```

```
   }, \
   { \
       "name": "quiesce", \
       "value": true || false \
   }, \
   { \
       "name": "memory", \
       "value": true || false \
   }, \
   { \
       "name": "description", \
       "value":${DESCRIPTION} # Optional \
   } \
   ]}' \
   "https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

POST method

BODY:

```
{
  "task_name": "snapshot",
  "parameters": [
    {
       "name": "snapshot_name",
       "value": "NAME"
    },
    {
       "name": "quiesce",
       "value": false
    },
    {
       "name": "memory",
       "value": false
    },
    {
       "name": "description",
       "value": "DESCRIPTION"
    }
   ]
  }
```

Note:

- The long ID in the URL after `iaasResources` is the RESOURCE_ID or the ID of the virtual machine.
- The description is an optional parameter.

Revert to a snapshot

This API reverts to the selected snapshot.

Note:

- The virtual machine `resource.id` can be retrieved from the update the state of the template instance API.
- The `snapshot_id` can be retrieved from the Get snapshots API.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{"task_name":"revertToSnapshot", "parameters": [ \
   { \
       "name": "snapshot_id", \
       "value": ${SNAPSHOT_ID} \
   } \
   ]}' \
   "https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

POST method

Sample BODY:

```
{
  "task_name": "revertToSnapshot",
  "parameters": [
    {
      "name": "snapshot_id",
```

```
      "value": "snapshot-19477"
    }
   ]
}
```

Note: The long ID in the URL after `iaasResources` is the RESOURCE_ID or the ID of the virtual machine.

Delete snapshot

This API deletes the selected snapshot.

Note:

- The virtual machine `resource.id` can be retrieved from the update the state of the template instance API.
- The `snapshot_id` can be retrieved from the Get snapshots API.

Sample URL:

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-H "Content-Type: application/json" -d '{"task_name":"deleteSnapshot", "parameters": [ \
   { \
       "name": "snapshot_id", \
       "value": ${SNAPSHOT_ID} \
   } \
   ]}' "https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

POST method

Sample BODY:

```
{
  "task_name": "deleteSnapshot",
  "parameters": [
   {
     "name": "snapshot_id",
     "value": "snapshot-19477"
   }
  ]
 }
```

Note: The long ID in the URL after `iaasResources` is the RESOURCE_ID or the ID of the virtual machine.

Revert to current snapshot

This API restores to the parent snapshot of the current state of a virtual machine.

Note: The virtual machine `resource.id` can be retrieved from the update the state of the template instance API.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-H "Content-Type: application/json" -d '{"task_name":"revertToCurrent"}' \
"https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Sample URL:

```
https://cam-proxy/cam/api/v1/iaasresources/5c47740ef76392001eb4d073/tasks
```

POST method

BODY:

```
{
  "task_name": "revertToCurrent"
}
```

Note: The long ID in the URL after `iaasResources` is the RESOURCE_ID or the ID of the virtual machine.

Get snapshots

This API retrieves a list of snapshots.

Note: The virtual machine **resource.id** can be retrieved from the update the state of the template instance API.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/query/getsnapshots?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Sample URL:

```
https://cam-proxy/cam/api/v1/iaasresources/5c47740ef76392001eb4d073/query/getsnapshots
```

GET method

Note: The long ID in the URL after **iaasResources** is the RESOURCE_ID or the ID of the virtual machine.

Taint a resource from a stack

Note: The virtual machine **resource.id** can be retrieved from the update the state of the template instance API.

```
export CAM_RESOURCE_ID=<resource.id>
```

Note: The **resource.id** can be retrieved from Update state of template instance POST /stacks/id/retrieve call.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${CAM_TEMPLATE_ID}/taint/${CAM_RESOURCE_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example output:

```
{
    "name": "vSphere95",
    "created_at": "2018-01-10T18:35:52.140Z",
    "templateId": "5a4fe8f98bec4000200dc310",
    "namespaceId": "default",
    "templateName": "SingleVirtualMachine",
    "templateCategory": "userCreated",
    "templateProvider": "IBM",
    "template_type": "Terraform",
    "applied_state_id": "5a565d4d8bec4000200dc32f",
    "latest_state_id": "5a56600a8bec4000200dc331",
    "cloud_connection_ids": [
        "5a3035908bec4000200dc30d"
    ],
    "id": "5a565d088bec4000200dc32c",
    "tenantId": "cd3ed377-8167-4078-b486-309b1f237928",
    "status": "IN_PROGRESS",
    "message": "",
    "action": "TAINT",
    "stack_id": "1616c4e0-f635-11e7-b2c8-2b0dbae0c060",
    "applied_status": "SUCCESS",
    "applied_action": "APPLY",
    "data": {
        "stack_id": "1616c4e0-f635-11e7-b2c8-2b0dbae0c060",
        "stack_job_id": "5a56600a035d8000204e8f27",
        "stack_job_is_latest": true,
        "action": "TAINT",
        "status": "IN_PROGRESS",
        "tenantId": "cd3ed377-8167-4078-b486-309b1f237928"
    }
}
```

Untaint a resource from a stack

Note: The virtual machine **resource.id** can be retrieved from the update the state of the template instance API.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks/${CAM_TEMPLATE_ID}/untaint/${CAM_RESOURCE_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example output:

```
{
    "name": "vSphere95",
```

```
    "created_at": "2018-01-10T18:35:52.140Z",
    "templateId": "5a4fe8f98bec4000200dc310",
    "namespaceId": "default",
    "templateName": "SingleVirtualMachine",
    "templateCategory": "userCreated",
    "templateProvider": "IBM",
    "template_type": "Terraform",
    "applied_state_id": "5a565d4d8bec4000200dc32f",
    "latest_state_id": "5a56618b8bec4000200dc332",
    "cloud_connection_ids": [
        "5a3035908bec4000200dc30d"
    ],
    "id": "5a565d088bec4000200dc32c",
    "tenantId": "cd3ed377-8167-4078-b486-309b1f237928",
    "status": "IN_PROGRESS",
    "message": "",
    "action": "UNTAINT",
    "stack_id": "1616c4e0-f635-11e7-b2c8-2b0dbae0c060",
    "applied_status": "SUCCESS",
    "applied_action": "APPLY",
    "data": {
        "stack_id": "1616c4e0-f635-11e7-b2c8-2b0dbae0c060",
        "stack_job_id": "5a56618b035d8000204e8f28",
        "stack_job_is_latest": true,
        "action": "UNTAINT",
        "status": "IN_PROGRESS",
        "tenantId": "cd3ed377-8167-4078-b486-309b1f237928"
    }
}
```

Retrieve and change power states for resources

This API retrieves and changes power states for resources.

Note: The virtual machine `resource.id` can be retrieved from the update the state of the template instance API.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/query/getPowerState?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

GET method

```
Response: {
"poweredOn": true || false
}
```

Start a virtual machine

Note: The virtual machine `resource.id` can be retrieved from the update the state of the template instance API.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{"task_name":"start"}' \
"https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example response

```
{
  "cloud_resource_id": "421ac557-7793-fdd5-7b83-c3525571abfe",
  "task_name": "start",
  "status": "IN_PROGRESS",
  "created": "2017-11-13T16:51:37.972Z",
  "updated": "2017-11-13T16:51:37.972Z",
  "stack_id": "7d55d130-c239-11e7-b95f-c5cc4d245984",
  "resource_id": "59ff272d4ba47e0020cb6583",
  "resource_name": "vsphere_virtual_machine.vm_1",
  "id": "5a09cd999fe5890020316468"
}
```

Stop a virtual machine

```
export CAM_RESOURCE_ID=<resource.id>
```

Note: The virtual machine **resource.id** can be retrieved from the update the state of the template instance API.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{"task_name":"stop"}' \
"https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example response

```
{
  "cloud_resource_id": "421ac557-7793-fdd5-7b83-c3525571abfe",
  "task_name": "stop",
  "status": "IN_PROGRESS",
  "created": "2017-11-13T16:47:58.298Z",
  "updated": "2017-11-13T16:47:58.298Z",
  "stack_id": "7d55d130-c239-11e7-b95f-c5cc4d245984",
  "resource_id": "59ff272d4ba47e0020cb6583",
  "resource_name": "vsphere_virtual_machine.vm_1",
  "id": "5a09ccbe9fe5890020316467"
}
```

Reset a virtual machine

You can reset and restart a VMware virtual machine.

Note: The virtual machine **resource.id** can be retrieved from the update the state of the template instance API.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{"task_name":"reset"}' \
"https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example response

```
{
  "cloud_resource_id": "421ac557-7793-fdd5-7b83-c3525571abfe",
  "task_name": "reset",
  "status": "IN_PROGRESS",
  "created": "2017-11-13T16:47:58.298Z",
  "updated": "2017-11-13T16:47:58.298Z",
  "stack_id": "7d55d130-c239-11e7-b95f-c5cc4d245984",
  "resource_id": "59ff272d4ba47e0020cb6583",
  "resource_name": "vsphere_virtual_machine.vm_1",
  "id": "5a09ccbe9fe5890020316467"
}
```

Get the status of a stopping or starting a virtual machine

```
export CAM_IAASTASK_ID=<response.id>
```

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" \
-d '{"task_name":"stop"}' \
"https://${TSA_HOSTNAME}/cam/api/v1/iaasresources/${CAM_RESOURCE_ID}/tasks/${CAM_IAASTASK_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example response:

```
{
  "cloud_resource_id": "421ac557-7793-fdd5-7b83-c3525571abfe",
  "task_name": "stop",
  "status": "SUCCESS",
  "created": "2017-11-13T16:47:58.298Z",
  "updated": "2017-11-13T16:48:03.728Z",
  "stack_id": "7d55d130-c239-11e7-b95f-c5cc4d245984",
  "resource_id": "59ff272d4ba47e0020cb6583",
  "resource_name": "vsphere_virtual_machine.vm_1",
  "message": "",
  "id": "5a09ccbe9fe5890020316467"
}
```

Retrieve all IaaS resource settings

This API allows you to get all Terraform resource settings.

```
curl -k -X GET \
  -H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
  "https://${TSA_HOSTNAME}/cam/api/v1/iaasresourcesettings?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Note: The cloudOE_spaceGuid query parameter is optional and the search results display all IaaS resource settings that `${NAMESPACE}` resolves to.

Retrieve a specific IaaS resource settings

This API allows you to show the details about a Terraform resource setting.

```
curl -k -X GET \
  -H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
 "https://${TSA_HOSTNAME}/cam/api/v1/iaasresourcesettings/${IAAS_RESOURCE_SETTINGS_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Create IaaS resource settings

This API allows an administrator to hide specific information about the Terraform resource.

```
curl -k -X POST \
 -H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
 -H "Content-Type: application/json" \
 "https://${TSA_HOSTNAME}/cam/api/v1/iaasresourcesettings?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}" \
 -d '{
     "type": "aws_virtual_machine",
     "attributes": ["cpu_reservation", "guest_ip_addresses"],
     "hidden": true
   }'
```

This example hides two attributes from all the `aws_virtual_machine` resources. In case you want to hide all the attributes from all the resource types, create an IaaS resource settings with type as `__all__`.

Request body parameters:

- `type` - the provider resource type (that is, vpshere_virtual_machine). This is not the resource name.
- `attributes` - list of attributes to be hidden. This parameter is optional. In case it is omitted, all the attributes for that specific resource type are hidden.
- `hidden` - It accepts true or false depending on whether you want to hide or show the attribute.

Note: The cloudOE_spaceGuid query parameter is optional and the IaaS resource setting is created in the namespace that ${NAMESPACE} resolves to.

Update IaaS resource settings

This API allows the cluster administrator and administrator to modify the settings for a Terraform resource type.

```
curl -k -X PUT \
  -H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
  -H "Content-Type: application/json" \
  -d '{"type": "aws_virtual_machine"}' \
 "https://${TSA_HOSTNAME}/cam/api/v1/iaasresourcesettings/${IAAS_RESOURCE_SETTINGS_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Delete IaaS resource settings

This API allows the cluster administrator and administrator to delete settings for a Terraform resource type.

```
curl -k -X DELETE \
 -H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
 "https://${TSA_HOSTNAME}/cam/api/v1/iaasresourcesettings/${IAAS_RESOURCE_SETTINGS_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Service and service instance APIs

This section includes all service and service instance APIs with examples.

- Service APIs
  - Create a service with default version
  - Add a service version to an existing service
  - Set a default version for a service
  - Edit a Service version
  - Delete a Service version in a service
  - Delete a Service (with all its versions)
  - GET the list of services
  - Get Service versions for a service using `service-id`
  - Get Service version manifest by service version ID
  - Publish a service version
  - Duplicate a service
  - Change the category of a service
  - Rename a service
  - Change service image for a service
  - Assign access to a service
  - Retire a service version
  - Unretire a service version
  - Auto-retire a service version
- Service Instance APIs
  - Deploy a Service Instance
  - Get the list of deployed instance: GET call
  - Get details of a Service Instance by using the id
  - Get input parameters of a Service Instance
  - Edit Service Instance
  - Retry Service Instance
  - Resume Service Instance
  - Upgrade or downgrade a Service Instance
  - Bind a service instance
  - Unbind Instance
  - Get custom actions
  - Invoke custom action
  - Terminate Service Instance
  - Cleanup Service Instance
  - Cleanup Service Instance in inprogress state
- Standalone action APIs
  - Create Standalone Action
  - Invoke Standalone Action
- ICON APIs
  - Upload service icons
  - Delete service icons
  - Retrieve all service icons
  - Download service icon

Service APIs

Create a service with default version

This API call creates a service version and adds a default version. The service and the service version are returned in response.

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/createService?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @service_create.json
```

Payload, content of 'service_create.json'

```
{
  "service": {
    "specVersion": "v3",
    "catalog_metadata": {
      "name": "SampleV1",
      "description": "A Sample Single Virtual Machine Service",
      "image": "serviceicon_1.svg",
      "category": "",
      "bullets": [
```

```
      ],
      "providerDisplayName": "IBM",
      "longDescription": "",
      "documentationUrl": "",
      "supportUrl": "",
      "bindable": false,
      "updatable": "",
      "systemTags": true,
      "author": "admin"
    },
    "tags": [
      {
        "name": "request_group",
        "label": "request_group",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The current context group id of the current user that requested the provisioning of
the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
        "isSystemTag": true,
        "default": "${svc_instance.group}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "request_user",
        "label": "request_user",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The user id of the current user that requested the provisioning of the template.",
        "isSystemTag": true,
        "default": "${svc_instance.owner}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_name",
        "label": "service_name",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Name provided for the service instance by the end user at time of request",
        "isSystemTag": true,
        "default": "${svc_instance.name}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_identifier",
        "label": "service_identifier",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
        "isSystemTag": true,
        "default": "${svc_instance.service_instance_id}",
        "customtype": "string",
        "permission": "Read-Only"
      }
    ],
    "actions": [
      {
        "type": "provision",
        "name": "Provision",
        "description": "Default action for deployment of service",
        "input_parameters": [
          {
            "name": "vmMemory",
            "label": "vmMemory",
            "customtype": "string",
```

```
            "type": "string",
            "immutable": false,
            "hidden": false,
            "required": false,
            "secured": false
          },
          {
            "name": "vmCpu",
            "label": "vmCpu",
            "customtype": "string",
            "type": "string",
            "immutable": false,
            "hidden": false,
            "required": false,
            "secured": false
          }
        ],
        "flow": {
          "conditions": [


          ],
          "templates": [
            {
              "SingleVirtualMachine": {
                "title": "SingleVirtualMachine_cca453",
                "template_name": "SingleVirtualMachine",
                "version": "2.3",
                "id": "singlevi23cd3149",
                "template_type": "Terraform",
                "template_content_type": "VMware vSphere",
                "template_provider": "VMware vSphere",
                "instance_name": "",
                "cloud_connection_name": "vmware",
                "template_data_objects": {


                },
                "template_params": {
                  "vm_1_name": "ver2vm",
                  "vm_1_dns_servers": [
                    "1.1.1.2"
                  ],
                  "vm_1_dns_suffixes": [
                    "test.cam.com"
                  ],
                  "vm_1_root_disk_keep_on_remove": "false",
                  "vm_1_domain": "test.cam.com",
                  "vm_1-image": "terraform/template/rhel_7_2_small",
                  "vm_1_root_disk_size": "25",
                  "vm_1_root_disk_controller_type": "scsi",
                  "vm_1_root_disk_datastore": "NSXcluster/datastore1",
                  "vm_1_root_disk_type": "eager_zeroed",
                  "vm_1_ipv4_gateway": "10.4.10.1",
                  "vm_1_ipv4_address": "10.4.10.17",
                  "vm_1_memory": "${input_parameters.vmMemory}",
                  "vm_1_ipv4_prefix_length": "23",
                  "vm_1_number_of_vcpu": "${input_parameters.vmCpu}",
                  "vm_1_network_interface_label": "external-net-410",
                  "vm_1_cluster": "icm-vmware-nsx-2",
                  "vm_1_datacenter": "NSX Manage To",
                  "vm_1_folder": "terraform/test",
                  "vm_1_adapter_type": "vmxnet3",
                  "vm_1_resource_pool": "cam"
                },
                "outputs": [
                ],
                "immutable": false
              }
            }
          ],
          "resthooks": [
          ],
          "notifications": [
          ],
          "sequence": {
            "0": "singlevi23cd3149"
          },
          "error_sequence": {
          }
        },
```

```
        "output_parameters": [
        ]
      }
    ],
    "plans": [
      {
        "name": "Standard",
        "description": "To deploy a Standard plan",
        "actions": [
        ],
        "plan_parameters": [
        ]
      }
    ]
  }
}
```

Response

```
{
  "id": "5df37c4aa051bd001d161161",
  "name": "SampleV1",
  "author": "admin",
  "image": "serviceicon_1.svg",
  "status": "",
  "serviceManifestId": "5df37c4ba051bd001d161162",
  "tag": "5c6e73077077fc001715703b",
  "creationTime": "2019-12-13T11:55:51.389Z",
  "lastUpdated": "2019-12-13T11:55:55.078Z",
  "namespaceId": "default",
  "tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
  "description": "A Sample Single Virtual Machine Service",
  "longDescription": "",
  "defaultServiceVersion": {
    "id": "5df37c4ba051bd001d161162",
    "version": "1.0.0.0",
    "ServiceEngine_InputFile": {
      "service": {
        "specVersion": "v3",
        "catalog_metadata": {
          "name": "SampleV1",
          "description": "A Sample Single Virtual Machine Service",
          "image": "serviceicon_1.svg",
          "category": "CloudServices",
          "bullets": [],
          "providerDisplayName": "IBM",
          "longDescription": "",
          "documentationUrl": "",
          "supportUrl": "",
          "bindable": false,
          "updatable": "",
          "systemTags": true,
          "author": "admin"
        },
```

```
        "tags": [
          {
            "name": "request_group",
            "label": "request_group",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The current context group id of the current user that requested the provisioning
of the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
            "isSystemTag": true,
            "default": "${svc_instance.group}",
            "customtype": "string",
            "permission": "Read-Only"
          },
```

```
          {
            "name": "request_user",
            "label": "request_user",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
```

```
            "description": "The user id of the current user that requested the provisioning of the
template.",
            "isSystemTag": true,
            "default": "${svc_instance.owner}",
            "customtype": "string",
            "permission": "Read-Only"
          },

          {
            "name": "service_name",
            "label": "service_name",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Name provided for the service instance by the end user at time of request",
            "isSystemTag": true,
            "default": "${svc_instance.name}",
            "customtype": "string",
            "permission": "Read-Only"
          },

          {
            "name": "service_identifier",
            "label": "service_identifier",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
            "isSystemTag": true,
            "default": "${svc_instance.service_instance_id}",
            "customtype": "string",
            "permission": "Read-Only"
          }

        ],
        "actions": [
          {
            "type": "provision",
            "name": "Provision",
            "description": "Default action for deployment of service",
            "input_parameters": [
              {
                "name": "vmMemory",
                "label": "vmMemory",
                "customtype": "string",
                "type": "string",
                "immutable": false,
                "hidden": false,
                "required": false,
                "secured": false
              },
              {
                "name": "vmCpu",
                "label": "vmCpu",
                "customtype": "string",
                "type": "string",
                "immutable": false,
                "hidden": false,
                "required": false,
                "secured": false
              }
            ],
            "flow": {
              "conditions": [],
              "templates": [
                {
                  "SingleVirtualMachine": {
                    "title": "SingleVirtualMachine_cca453",
                    "template_name": "SingleVirtualMachine",
                    "version": "2.3",
                    "id": "singlevi23cd3149",
                    "template_type": "Terraform",
                    "template_content_type": "VMware vSphere",
                    "template_provider": "VMware vSphere",
                    "instance_name": "",
                    "cloud_connection_name": "vmware",
```

```
                    "template_data_objects": {},
                    "template_params": {
                      "vm_1_name": "ver2vm",
                      "vm_1_dns_servers": [
                        "1.1.1.2"
                      ],
                      "vm_1_dns_suffixes": [
                        "test.cam.com"
                      ],
                      "vm_1_root_disk_keep_on_remove": "false",
                      "vm_1_domain": "test.cam.com",
                      "vm_1-image": "terraform/template/rhel_7_2_small",
                      "vm_1_root_disk_size": "25",
                      "vm_1_root_disk_controller_type": "scsi",
                      "vm_1_root_disk_datastore": "NSXcluster/datastore1",
                      "vm_1_root_disk_type": "eager_zeroed",
                      "vm_1_ipv4_gateway": "10.4.10.1",
                      "vm_1_ipv4_address": "10.4.10.17",
                      "vm_1_memory": "${input_parameters.vmMemory}",
                      "vm_1_ipv4_prefix_length": "23",
                      "vm_1_number_of_vcpu": "${input_parameters.vmCpu}",
                      "vm_1_network_interface_label": "external-net-410",
                      "vm_1_cluster": "icm-vmware-nsx-2",
                      "vm_1_datacenter": "NSX Manage To",
                      "vm_1_folder": "terraform/test",
                      "vm_1_adapter_type": "vmxnet3",
                      "vm_1_resource_pool": "cam"
                    },

                    "outputs": [],
                    "immutable": false
                  }
                }
              ],

              "resthooks": [],
              "notifications": [],
              "sequence": {
                "0": "singlevi23cd3149"
              },
              "error_sequence": {}
            },
            "output_parameters": []
          }
        ],
        "plans": [
          {
            "name": "Standard",
            "description": "To deploy a Standard plan",
            "actions": [],
            "plan_parameters": []
          }
        ],
        "version": "1.0.0.0"
      }
    },
    "service_id": "5df37c4aa051bd001d161161",
    "namespaceId": "default",
    "author": "admin",
    "specVersion": "v3",
    "status": "Draft",
    "tags": [
      {
        "name": "request_group",
        "label": "request_group",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The current context group id of the current user that requested the provisioning of
the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
        "isSystemTag": true,
        "default": "${svc_instance.group}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "request_user",
        "label": "request_user",
```

```
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The user id of the current user that requested the provisioning of the template.",
        "isSystemTag": true,
        "default": "${svc_instance.owner}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_name",
        "label": "service_name",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Name provided for the service instance by the end user at time of request",
        "isSystemTag": true,
        "default": "${svc_instance.name}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_identifier",
        "label": "service_identifier",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
        "isSystemTag": true,
        "default": "${svc_instance.service_instance_id}",
        "customtype": "string",
        "permission": "Read-Only"
      }
    ],
    "systemTags": true,
    "creationTime": "2019-12-13T11:55:54.970Z"
  },
  "warning": [
    "No 'Category' is provided in the source code. Assigning the service to 'CloudServices' \n",
    "No 'version' is provided in the source code. Assigning the service, the default version '1.0.0.0' \n"
  ]
}
```

Add a service version to an existing service

Service name, category, and image cannot be changed by using this API.

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICE_ID}/addServiceVersion?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
 -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
 -H 'Content-Type: application/json' \
 -H 'cache-control: no-cache' \
 -d @service_version_add.json
```

Payload: Content of `service_version_add.json`

```
{
 "service": {
  "version": "1.0.0.1",
  "specVersion": "v3",
  "catalog_metadata": {
   "name": "SampleV1",
   "description": "A Sample Single Virtual Machine Service",
   "image": "serviceicon_1.svg",
   "category": "CloudServices",
   "bullets": [],
   "providerDisplayName": "IBM",
   "longDescription": "",
   "documentationUrl": "",
   "supportUrl": "",
   "bindable": false,
   "updatable": "",
   "systemTags": true,
```

```
   "author": "admin"
  },
  "tags": [
   {
    "name": "request_group",
    "label": "request_group",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "The current context group id of the current user that requested the provisioning of the
template. This is important because a user may be part of more than one group and this identifies the context
of which group the user made the request.",
    "isSystemTag": true,
    "default": "${svc_instance.group}",
    "customtype": "string",
    "permission": "Read-Only"
   },
   {
    "name": "request_user",
    "label": "request_user",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "The user id of the current user that requested the provisioning of the template.",
    "isSystemTag": true,
    "default": "${svc_instance.owner}",
    "customtype": "string",
    "permission": "Read-Only"
   },
   {
    "name": "service_name",
    "label": "service_name",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "Name provided for the service instance by the end user at time of request",
    "isSystemTag": true,
    "default": "${svc_instance.name}",
    "customtype": "string",
    "permission": "Read-Only"
   },
   {
    "name": "service_identifier",
    "label": "service_identifier",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
    "isSystemTag": true,
    "default": "${svc_instance.service_instance_id}",
    "customtype": "string",
    "permission": "Read-Only"
   }
  ],
  "actions": [
   {
    "type": "provision",
    "name": "Provision",
    "description": "Default action for deployment of service",
    "input_parameters": [
     {
      "name": "vmMemory",
      "label": "vmMemory",
      "customtype": "string",
      "type": "string",
      "immutable": false,
      "hidden": false,
      "required": false,
      "secured": false
     },
     {
      "name": "vmCpu",
```

```
     "label": "vmCpu",
     "customtype": "string",
     "type": "string",
     "immutable": false,
     "hidden": false,
     "required": false,
     "secured": false
    }
   ],
   "flow": {
    "conditions": [],
    "templates": [
     {
      "SingleVirtualMachine": {
       "title": "SingleVirtualMachine_cca453",
       "template_name": "SingleVirtualMachine",
       "version": "2.3",
       "id": "singlevi23cd3149",
       "template_type": "Terraform",
       "template_content_type": "VMware vSphere",
       "template_provider": "VMware vSphere",
       "instance_name": "",
       "cloud_connection_name": "vmware",
       "template_data_objects": {},
       "template_params": {
        "vm_1_name": "ver2vm",
        "vm_1_dns_servers": [
         "1.1.1.2"
        ],
        "vm_1_dns_suffixes": [
         "test.cam.com"
        ],
        "vm_1_root_disk_keep_on_remove": "false",
        "vm_1_domain": "test.cam.com",
        "vm_1-image": "terraform/template/rhel_7_2_small",
        "vm_1_root_disk_size": "25",
        "vm_1_root_disk_controller_type": "scsi",
        "vm_1_root_disk_datastore": "NSXcluster/datastore1",
        "vm_1_root_disk_type": "eager_zeroed",
        "vm_1_ipv4_gateway": "10.4.10.1",
        "vm_1_ipv4_address": "10.4.10.17",
        "vm_1_memory": "${input_parameters.vmMemory}",
        "vm_1_ipv4_prefix_length": "23",
        "vm_1_number_of_vcpu": "${input_parameters.vmCpu}",
        "vm_1_network_interface_label": "external-net-410",
        "vm_1_cluster": "icm-vmware-nsx-2",
        "vm_1_datacenter": "NSX Manage To",
        "vm_1_folder": "terraform/test",
        "vm_1_adapter_type": "vmxnet3",
        "vm_1_resource_pool": "cam"
       },
       "outputs": [],
       "immutable": false
      }
     }
    ],
    "resthooks": [],
    "notifications": [
     {
      "emailnotification": {
       "title": "emailnotification_4278ce",
       "type": "email",
       "receiverList": "testmail@testr-mail.net",
       "id": "emailnot21410720",
       "sender": "CAMadmin",
       "copyToList": "",
       "subject": "VM Creation Succeful",
       "body": "VM create at ${templates.singlevi23cd3149.created_at}",
       "immutable": false
      }
     }
    ],
    "sequence": {
     "0": "singlevi23cd3149",
     "1": "emailnot21410720"
    },
    "error_sequence": {}
   },
   "output_parameters": []
  }
 ],
```

```
  "plans": [
   {
    "name": "Standard",
    "description": "To deploy a Standard plan",
    "actions": [],
    "plan_parameters": []
   }
  ]
 }
}
```

Response

```
{
  "id": "5df37f91a051bd001d161171",
  "version": "1.0.0.1",
  "ServiceEngine_InputFile": {
    "service": {
      "version": "1.0.0.1",
      "specVersion": "v3",
      "catalog_metadata": {
        "name": "SampleV1",
        "description": "A Sample Single Virtual Machine Service",
        "image": "serviceicon_1.svg",
        "category": "CloudServices",
        "bullets": [],
        "providerDisplayName": "IBM",
        "longDescription": "",
        "documentationUrl": "",
        "supportUrl": "",
        "bindable": false,
        "updatable": "",
        "systemTags": true,
        "author": "admin"
      },
      "tags": [
        {
          "name": "request_group",
          "label": "request_group",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "The current context group id of the current user that requested the provisioning of
the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
          "isSystemTag": true,
          "default": "${svc_instance.group}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
          "name": "request_user",
          "label": "request_user",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "The user id of the current user that requested the provisioning of the template.",
          "isSystemTag": true,
          "default": "${svc_instance.owner}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
          "name": "service_name",
          "label": "service_name",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "Name provided for the service instance by the end user at time of request",
          "isSystemTag": true,
          "default": "${svc_instance.name}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
```

```
          "name": "service_identifier",
          "label": "service_identifier",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
          "isSystemTag": true,
          "default": "${svc_instance.service_instance_id}",
          "customtype": "string",
          "permission": "Read-Only"
        }
      ],
      "actions": [
        {
          "type": "provision",
          "name": "Provision",
          "description": "Default action for deployment of service",
          "input_parameters": [
            {
              "name": "vmMemory",
              "label": "vmMemory",
              "customtype": "string",
              "type": "string",
              "immutable": false,
              "hidden": false,
              "required": false,
              "secured": false
            },
            {
              "name": "vmCpu",
              "label": "vmCpu",
              "customtype": "string",
              "type": "string",
              "immutable": false,
              "hidden": false,
              "required": false,
              "secured": false
            }
          ],
          "flow": {
            "conditions": [],
            "templates": [
              {
                "SingleVirtualMachine": {
                  "title": "SingleVirtualMachine_cca453",
                  "template_name": "SingleVirtualMachine",
                  "version": "2.3",
                  "id": "singlevi23cd3149",
                  "template_type": "Terraform",
                  "template_content_type": "VMware vSphere",
                  "template_provider": "VMware vSphere",
                  "instance_name": "",
                  "cloud_connection_name": "vmware",
                  "template_data_objects": {},
                  "template_params": {
                    "vm_1_name": "ver2vm",
                    "vm_1_dns_servers": [
                      "1.1.1.2"
                    ],
                    "vm_1_dns_suffixes": [
                      "test.cam.com"
                    ],
                    "vm_1_root_disk_keep_on_remove": "false",
                    "vm_1_domain": "test.cam.com",
                    "vm_1-image": "terraform/template/rhel_7_2_small",
                    "vm_1_root_disk_size": "25",
                    "vm_1_root_disk_controller_type": "scsi",
                    "vm_1_root_disk_datastore": "NSXcluster/datastore1",
                    "vm_1_root_disk_type": "eager_zeroed",
                    "vm_1_ipv4_gateway": "10.4.10.1",
                    "vm_1_ipv4_address": "10.4.10.17",
                    "vm_1_memory": "${input_parameters.vmMemory}",
                    "vm_1_ipv4_prefix_length": "23",
                    "vm_1_number_of_vcpu": "${input_parameters.vmCpu}",
                    "vm_1_network_interface_label": "external-net-410",
                    "vm_1_cluster": "icm-vmware-nsx-2",
                    "vm_1_datacenter": "NSX Manage To",
                    "vm_1_folder": "terraform/test",
```

```
                  "vm_1_adapter_type": "vmxnet3",
                  "vm_1_resource_pool": "cam"
                },
                "outputs": [],
                "immutable": false
              }
            }
          ],
          "resthooks": [],
          "notifications": [
            {
              "emailnotification": {
                "title": "emailnotification_4278ce",
                "type": "email",
                "receiverList": "testmail@testr-mail.net",
                "id": "emailnot21410720",
                "sender": "CAMadmin",
                "copyToList": "",
                "subject": "VM Creation Succeful",
                "body": "VM create at ${templates.singlevi23cd3149.created_at}",
                "immutable": false
              }
            }
          ],
          "sequence": {
            "0": "singlevi23cd3149",
            "1": "emailnot21410720"
          },
          "error_sequence": {}
        },
        "output_parameters": []
      }
    ],
    "plans": [
      {
        "name": "Standard",
        "description": "To deploy a Standard plan",
        "actions": [],
        "plan_parameters": []
      }
    ]
  }
},
"service_id": "5df37f85a051bd001d16116d",
"tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
"namespaceId": "default",
"author": "admin",
"specVersion": "v3",
"status": "Draft",
"tags": [
  {
    "name": "request_group",
    "label": "request_group",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "The current context group id of the current user that requested the provisioning of the
template. This is important because a user may be part of more than one group and this identifies the context
of which group the user made the request.",
    "isSystemTag": true,
    "default": "${svc_instance.group}",
    "customtype": "string",
    "permission": "Read-Only"
  },
  {
    "name": "request_user",
    "label": "request_user",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "The user id of the current user that requested the provisioning of the template.",
    "isSystemTag": true,
    "default": "${svc_instance.owner}",
    "customtype": "string",
    "permission": "Read-Only"
  },
  {
```

```
      "name": "service_name",
      "label": "service_name",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "Name provided for the service instance by the end user at time of request",
      "isSystemTag": true,
      "default": "${svc_instance.name}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "service_identifier",
      "label": "service_identifier",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
      "isSystemTag": true,
      "default": "${svc_instance.service_instance_id}",
      "customtype": "string",
      "permission": "Read-Only"
    }
  ],
  "systemTags": true,
  "creationTime": "2019-12-13T12:09:53.433Z",
  "warning": []
}
```

Set a default version for a service

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICE_ID}/updateServiceDetails?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @service_default_version.json
```

Payload sample1:

```
"default_version_id": "<<version id>>"
```

Payload sample2:

```
{
 "default_version_name": "1.0.0.1"
}
```

Response:

```
{
  "service": {
    "id": "5df38f28a051bd001d16117d",
    "name": "SampleV1",
    "serviceManifestId": "5df38f30a051bd001d161181",
    "author": "admin",
    "image": "serviceicon_1.svg",
    "tag": "5c6e73077077fc001715703b",
    "creationTime": "2019-12-13T13:16:23.237Z",
    "lastUpdated": "2019-12-13T13:25:01.646Z",
    "namespaceId": "default",
    "tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
    "description": "A Sample Single Virtual Machine Service",
    "longDescription": ""
  }
}
```

Edit a Service Version

This API edits a service version. This API cannot change service name, category, and image.

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/${SERVICE_VERSION_ID}/updateServiceVersion?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
 -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
```

```
 -H 'Content-Type: application/json' \
 -H 'cache-control: no-cache' \
 -d @service_version_edit.json
```

Payload: content of service_version_edit.json

```
{
 "service": {
  "version": "1.0.0.1",
  "specVersion": "v3",
  "catalog_metadata": {
   "name": "SampleV1",
   "description": "A Sample Single Virtual Machine Service",
   "image": "serviceicon_1.svg",
   "category": "CloudServices",
   "bullets": [],
   "providerDisplayName": "IBM",
   "longDescription": "",
   "documentationUrl": "",
   "supportUrl": "",
   "bindable": false,
   "updatable": "",
   "systemTags": true,
   "author": "admin"
  },
  "tags": [
   {
    "name": "request_group",
    "label": "request_group",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "The current context group id of the current user that requested the provisioning of the
template. This is important because a user may be part of more than one group and this identifies the context
of which group the user made the request.",
    "isSystemTag": true,
    "default": "${svc_instance.group}",
    "customtype": "string",
    "permission": "Read-Only"
   },
   {
    "name": "request_user",
    "label": "request_user",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "The user id of the current user that requested the provisioning of the template.",
    "isSystemTag": true,
    "default": "${svc_instance.owner}",
    "customtype": "string",
    "permission": "Read-Only"
   },
   {
    "name": "service_name",
    "label": "service_name",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "Name provided for the service instance by the end user at time of request",
    "isSystemTag": true,
    "default": "${svc_instance.name}",
    "customtype": "string",
    "permission": "Read-Only"
   },
   {
    "name": "service_identifier",
    "label": "service_identifier",
    "type": "string",
    "immutable": true,
    "hidden": false,
    "required": false,
    "secured": false,
    "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
    "isSystemTag": true,
```

```
    "default": "${svc_instance.service_instance_id}",
    "customtype": "string",
    "permission": "Read-Only"
   }
  ],
  "actions": [
   {
    "type": "provision",
    "name": "Provision",
    "description": "Default action for deployment of service",
    "input_parameters": [
     {
      "name": "vmMemory",
      "label": "vmMemory",
      "customtype": "string",
      "type": "string",
      "immutable": false,
      "hidden": false,
      "required": false,
      "secured": false
     },
     {
      "name": "vmCpu",
      "label": "vmCpu",
      "customtype": "string",
      "type": "string",
      "immutable": false,
      "hidden": false,
      "required": false,
      "secured": false
     }
    ],
    "flow": {
     "conditions": [],
     "templates": [
      {
       "SingleVirtualMachine": {
        "title": "SingleVirtualMachine_cca453",
        "template_name": "SingleVirtualMachine",
        "version": "2.3",
        "id": "singlevi23cd3149",
        "template_type": "Terraform",
        "template_content_type": "VMware vSphere",
        "template_provider": "VMware vSphere",
        "instance_name": "",
        "cloud_connection_name": "vmware",
        "template_data_objects": {},
        "template_params": {
         "vm_1_name": "ver2vm",
         "vm_1_dns_servers": [
          "1.1.1.2"
         ],
         "vm_1_dns_suffixes": [
          "test.cam.com"
         ],
         "vm_1_root_disk_keep_on_remove": "false",
         "vm_1_domain": "test.cam.com",
         "vm_1-image": "terraform/template/rhel_7_2_small",
         "vm_1_root_disk_size": "25",
         "vm_1_root_disk_controller_type": "scsi",
         "vm_1_root_disk_datastore": "NSXcluster/datastore1",
         "vm_1_root_disk_type": "eager_zeroed",
         "vm_1_ipv4_gateway": "10.4.10.1",
         "vm_1_ipv4_address": "10.4.10.17",
         "vm_1_memory": "${input_parameters.vmMemory}",
         "vm_1_ipv4_prefix_length": "23",
         "vm_1_number_of_vcpu": "${input_parameters.vmCpu}",
         "vm_1_network_interface_label": "external-net-410",
         "vm_1_cluster": "icm-vmware-nsx-2",
         "vm_1_datacenter": "NSX Manage To",
         "vm_1_folder": "terraform/test",
         "vm_1_adapter_type": "vmxnet3",
         "vm_1_resource_pool": "cam"
        },
        "outputs": [],
        "immutable": false,
        "provider_engine_version": "0.11.11"
       }
      }
     ],
     "resthooks": [],
```

```
    "notifications": [
     {
      "emailnotification": {
       "title": "emailnotification_4278ce",
       "type": "email",
       "receiverList": "testmail@test-mail.sample.net",
       "id": "emailnot21410720",
       "sender": "CAMadmin",
       "copyToList": "",
       "subject": "VM Creation Succeful",
       "body": "VM create at ${templates.singlevi23cd3149.created_at}",
       "immutable": false
      }
     },
     {
      "emailnotification": {
       "title": "emailnotification_ac336e",
       "type": "email",
       "receiverList": "testmail@test-mail.sample.net",
       "id": "emailnotd384eeec",
       "sender": "CAMadmin",
       "copyToList": "",
       "subject": "VM Creation failed",
       "body": "VM creation failure caused by : ${svc_instance.error.message}",
       "immutable": false
      }
     }
    ],
    "sequence": {
     "0": "singlevi23cd3149",
     "1": "emailnot21410720"
    },
    "error_sequence": {
     "0": "emailnotd384eeec"
    }
   },
   "output_parameters": []
  }
 ],
 "plans": [
  {
   "name": "Standard",
   "description": "To deploy a Standard plan",
   "actions": [],
   "plan_parameters": []
  }
 ]
}
}
```

Response:

```
{
  "id": "5df38f30a051bd001d161181",
  "version": "1.0.0.1",
  "ServiceEngine_InputFile": {
    "service": {
      "version": "1.0.0.1",
      "specVersion": "v3",
      "catalog_metadata": {
        "name": "SampleV1",
        "description": "A Sample Single Virtual Machine Service",
        "image": "serviceicon_1.svg",
        "category": "CloudServices",
        "bullets": [],
        "providerDisplayName": "IBM",
        "longDescription": "",
        "documentationUrl": "",
        "supportUrl": "",
        "bindable": false,
        "updatable": "",
        "systemTags": true,
        "author": "admin"
      },
      "tags": [
        {
          "name": "request_group",
          "label": "request_group",
          "type": "string",
          "immutable": true,
          "hidden": false,
```

```
          "required": false,
          "secured": false,
          "description": "The current context group id of the current user that requested the provisioning of
the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
          "isSystemTag": true,
          "default": "${svc_instance.group}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
          "name": "request_user",
          "label": "request_user",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "The user id of the current user that requested the rovisioning of the template.",
          "isSystemTag": true,
          "default": "${svc_instance.owner}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
          "name": "service_name",
          "label": "service_name",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "Name provided for the service instance by the end user at time of request",
          "isSystemTag": true,
          "default": "${svc_instance.name}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
          "name": "service_identifier",
          "label": "service_identifier",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
          "isSystemTag": true,
          "default": "${svc_instance.service_instance_id}",
          "customtype": "string",
          "permission": "Read-Only"
        }
      ],
      "actions": [
        {
          "type": "provision",
          "name": "Provision",
          "description": "Default action for deployment of service",
          "input_parameters": [
            {
              "name": "vmMemory",
              "label": "vmMemory",
              "customtype": "string",
              "type": "string",
              "immutable": false,
              "hidden": false,
              "required": false,
              "secured": false
            },
            {
              "name": "vmCpu",
              "label": "vmCpu",
              "customtype": "string",
              "type": "string",
              "immutable": false,
              "hidden": false,
              "required": false,
              "secured": false
            }
          ],
```

```
          "flow": {
            "conditions": [],
            "templates": [
              {
                "SingleVirtualMachine": {
                  "title": "SingleVirtualMachine_cca453",
                  "template_name": "SingleVirtualMachine",
                  "version": "2.3",
                  "id": "singlevi23cd3149",
                  "template_type": "Terraform",
                  "template_content_type": "VMware vSphere",
                  "template_provider": "VMware vSphere",
                  "instance_name": "",
                  "cloud_connection_name": "vmware",
                  "template_data_objects": {},
                  "template_params": {
                    "vm_1_name": "ver2vm",
                    "vm_1_dns_servers": [
                      "1.1.1.2"
                    ],
                    "vm_1_dns_suffixes": [
                      "test.cam.com"
                    ],
                    "vm_1_root_disk_keep_on_remove": "false",
                    "vm_1_domain": "test.cam.com",
                    "vm_1-image": "terraform/template/rhel_7_2_small",
                    "vm_1_root_disk_size": "25",
                    "vm_1_root_disk_controller_type": "scsi",
                    "vm_1_root_disk_datastore": "NSXcluster/datastore1",
                    "vm_1_root_disk_type": "eager_zeroed",
                    "vm_1_ipv4_gateway": "10.4.10.1",
                    "vm_1_ipv4_address": "10.4.10.17",
                    "vm_1_memory": "${input_parameters.vmMemory}",
                    "vm_1_ipv4_prefix_length": "23",
                    "vm_1_number_of_vcpu": "${input_parameters.vmCpu}",
                    "vm_1_network_interface_label": "external-net-410",
                    "vm_1_cluster": "icm-vmware-nsx-2",
                    "vm_1_datacenter": "NSX Manage To",
                    "vm_1_folder": "terraform/test",
                    "vm_1_adapter_type": "vmxnet3",
                    "vm_1_resource_pool": "cam"
                  },
                  "outputs": [],
                  "immutable": false,
                  "provider_engine_version": "0.11.11"
                }
              }
            ],
            "resthooks": [],
            "notifications": [
              {
                "emailnotification": {
                  "title": "emailnotification_4278ce",
                  "type": "email",
                  "receiverList": "testmail@test-mail.sample.net",
                  "id": "emailnot21410720",
                  "sender": "CAMadmin",
                  "copyToList": "",
                  "subject": "VM Creation Succeful",
                  "body": "VM create at ${templates.singlevi23cd3149.created_at}",
                  "immutable": false
                }
              },
              {
                "emailnotification": {
                  "title": "emailnotification_ac336e",
                  "type": "email",
                  "receiverList": "testmail@test-mail.sample.net",
                  "id": "emailnotd384eeec",
                  "sender": "CAMadmin",
                  "copyToList": "",
                  "subject": "VM Creation failed",
                  "body": "VM creation failure caused by : ${svc_instance.error.message}",
                  "immutable": false
                }
              }
            ],
            "sequence": {
              "0": "singlevi23cd3149",
              "1": "emailnot21410720"
            },
```

```
            "error_sequence": {
              "0": "emailnotd384eeec"
            }
          },
          "output_parameters": []
        }
      ],
      "plans": [
        {
          "name": "Standard",
          "description": "To deploy a Standard plan",
          "actions": [],
          "plan_parameters": []
        }
      ]
    }
  },
  "service_id": "5df38f28a051bd001d16117d",
  "tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
  "namespaceId": "default",
  "author": "admin",
  "specVersion": "v3",
  "status": "Draft",
  "tags": [
    {
      "name": "request_group",
      "label": "request_group",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "The current context group id of the current user that requested the provisioning of the
template. This is important because a user may be part of more than one group and this identifies the context
of which group the user made the request.",
      "isSystemTag": true,
      "default": "${svc_instance.group}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "request_user",
      "label": "request_user",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "The user id of the current user that requested the provisioning of the template.",
      "isSystemTag": true,
      "default": "${svc_instance.owner}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "service_name",
      "label": "service_name",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "Name provided for the service instance by the end user at time of request",
      "isSystemTag": true,
      "default": "${svc_instance.name}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "service_identifier",
      "label": "service_identifier",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
      "isSystemTag": true,
      "default": "${svc_instance.service_instance_id}",
      "customtype": "string",
```

```
      "permission": "Read-Only"
    }
  ],
  "systemTags": true,
  "creationTime": "2019-12-13T13:16:32.250Z",
  "serviceInstanceCount": 0,
  "warning": []
}
```

Delete a Service version in a service

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICE_ID}/deleteService?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @service_version_delete.json
```

Payload sample1:

```
{
  "version_id": "{{VERION_ID}}"
}
```

Payload sample2:

```
{
  "version": "{{VERSION}}"
}
```

Delete a Service (with all its versions)

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICE_ID}/deleteService?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @service_delete.json
```

Payload

```
 {}
```

Response

```
{
  "count": "1"
}
```

GET the list of services

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" "https://${TSA_HOSTNAME}/cam/composer/api/v1/service?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

The service records that belong to the namespace value provided for **cloudOE_spaceGuid** are fetched as a result of this curl command. If you specify any value for **cloudOE_spaceGuid**, then all default services are fetched along with services that belong to globally accessible namespace.

Example response:

```
[{
  "name": "helmmultiple",
  "author": "admin",
  "image": "serviceicon_1.svg",
  "specVersion": "v2",
  "status": "Draft",
  "plans": {
    "Standard": "5ab0dcb27bf798001e85724b"
  },
  "tags": [],
  "parameters": [],
  "serviceManifestId": "5ab0dcb27bf798001e857249",
  "lastUpdated": "2018-03-20T10:04:34.051Z",
  "published": "",
  "creationTime": "Tue Mar 20 2018 10:04:34 GMT+0000 (UTC)",
  "conditions": [{
    "ibm-charts/ibm-nodejs-sample": {
```

```
    "title": "ibm-charts/ibm-nodejs-sample",
    "template_name": "ibm-nodejs-sample",
    "version": "1.2.0",
    "id": "ibmchartc99c9c4a",
    "template_type": "Helm",
    "template_content_type": "Cloud Pak for Multicloud Management",
    "template_provider": "Cloud Pak for Multicloud Management",
    "instance_name": "",
    "cloud_connection_name": "MCM Connection",
    "template_dataobjects": {},
    "template_params": {
    "namespace": "default",
      "image-repository": "ibmcom/icp-nodejs-sample",
      "image-tag": "latest",
      "image-pullPolicy": "Always",
      "livenessProbe-initialDelaySeconds": 30,
      "livenessProbe-periodSeconds": 10,
      "service-name": "Node",
      "service-type": "NodePort",
      "service-port": 3000,
      "resources-limits-cpu": "100m",
      "resources-limits-memory": "128Mi",
      "resources-requests-cpu": "100m",
      "resources-requests-memory": "128Mi",
      "autoscaling-enabled": "",
      "autoscaling-minReplicas": 1,
      "autoscaling-maxReplicas": 2,
      "autoscaling-metrics-cpu-targetAverageUtilization": "100m"
    },
    "outputs": []
    }
  }, {
    "publichelm/wordpress": {
      "title": "publichelm/wordpress",
      "template_name": "wordpress",
      "version": "0.8.8",
      "id": "publichea9b36841",
      "template_type": "Helm",
      "template_content_type": "Cloud Pak for Multicloud Management",
      "template_provider": "Cloud Pak for Multicloud Management",
      "instance_name": "",
      "cloud_connection_name": "MCM Connection",
      "template_dataobjects": {},
      "template_params": {
        "namespace": "services",
        "image": "bitnami/wordpress:4.9.4-r2",
        "imagePullPolicy": "IfNotPresent",
        "wordpressUsername": "user",
        "wordpressEmail": "user@example.com",
        "wordpressFirstName": "FirstName",
        "wordpressLastName": "LastName",
        "wordpressBlogName": "User's Blog!",
        "allowEmptyPassword": "yes",
        "externalDatabase-host": "localhost",
        "externalDatabase-user": "bn_wordpress",
        "externalDatabase-password": "",
        "externalDatabase-database": "bitnami_wordpress",
        "externalDatabase-port": 3306,
                                 "mariadb-enabled": true,
                                 "mariadb-mariadbDatabase": "bitnami_wordpress",
                                 "mariadb-mariadbUser": "bn_wordpress",
                                 "mariadb-persistence-enabled": true,
                                 "mariadb-persistence-accessMode": "ReadWriteOnce",
                                 "mariadb-persistence-size": "8Gi",
                                 "serviceType": "LoadBalancer",
                                 "healthcheckHttps": "",
                                 "livenessProbe-initialDelaySeconds": 120,
                                 "livenessProbe-periodSeconds": 10,
                                 "livenessProbe-timeoutSeconds": 5,
                                 "livenessProbe-failureThreshold": 6,
                                 "livenessProbe-successThreshold": 1,
                                 "readinessProbe-initialDelaySeconds": 30,
                                 "readinessProbe-periodSeconds": 10,
                                 "readinessProbe-timeoutSeconds": 5,
                                 "readinessProbe-failureThreshold": 6,
                                 "readinessProbe-successThreshold": 1,
                                 "ingress-enabled": "",
                                 "ingress-hosts-0-name": "wordpress.local",
                                 "ingress-hosts-0-tls": "",
                                 "ingress-hosts-0-tlsSecret": "wordpress.local-tls",
                                 "ingress-hosts-0-annotations": "",
```

```
                                "ingress-secrets": "",
                                "persistence-enabled": true,
                                "persistence-accessMode": "ReadWriteOnce",
                                "persistence-size": "10Gi",
                                "resources-requests-memory": "512Mi",
                                "resources-requests-cpu": "300m"
                        },
                        "outputs": []
                }
        }, {
                "publichelm/filebeat": {
                        "title": "publichelm/filebeat",
                        "template_name": "filebeat",
                        "version": "0.1.0",
                        "id": "publiche046d1a27",
                        "template_type": "Helm",
                        "template_content_type": "Cloud Pak for Multicloud Management",
                        "template_provider": "Cloud Pak for Multicloud Management",
                        "instance_name": "",
                        "cloud_connection_name": "MCM Connection",
                        "template_dataobjects": {},
                        "template_params": {
                                "namespace": "kube-system",
                                "image-repository": "docker.elastic.co/beats/filebeat",
                                "image-tag": "6.2.2",
                                "image-pullPolicy": "IfNotPresent",
                                "config-processors-0-add_cloud_metadata": "",
                                "rbac-create": true,
                                "serviceAccount-create": true,
                                "serviceAccount-name": "",
                                "config-filebeat.config-prospectors-path":
"${path.config}/prospectors.d/*.yml",
                                "config-filebeat.config-prospectors-reload.enabled": "",
                                "config-filebeat.config-modules-path": "${path.config}/modules.d/*.yml",
                                "config-filebeat.config-modules-reload.enabled": "",
                                "config-filebeat.prospectors-0-type": "log",
                                "config-filebeat.prospectors-0-enabled": true,
                                "config-filebeat.prospectors-0-paths-0": "/var/log/**",
                                "config-filebeat.prospectors-0-exclude_files-0": "(\\.[0-9]{1,})$",
                                "config-filebeat.prospectors-0-exclude_files-1":
"templates/clusterrolebinding.yaml'.gz$'",
                                "config-output.file-path": "/usr/share/filebeat/data",
                                "config-output.file-filename": "filebeat",
                                "config-output.file-rotate_every_kb": 10000,
                                "config-output.file-number_of_files": 5
                        },
                        "outputs": []
                }
        }],
        "tag": "5a8e69872210e0001d311d9a",
        "template_mapping": {
                "ibmchartc99c9c4a": "ibm-nodejs-sample",
                "publichea9b36841": "wordpress",
                "publiche046d1a27": "filebeat"
        },
        "id": "5ab0dcb27bf798001e85724a",
        "orgGuid": "dummy-org-id",
        "spaceGuid": "dummy-space-id",
        "namespaceId": "dummy-space-id",
        "displayName": "helmmultiple"
}]
```


Get Service versions for a service by using `service-id`

This API fetches all the service versions for a specific service by using the service ID.

```
curl -k -X GET \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICEID}/ServiceVersions?
filter[order]=updated_time%20DESC&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENAN
T_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache'
```

Response:


```
[{"version":"1.0.0.0","ServiceEngine_InputFile":{"service":{"specVersion":"v3","catalog_metadata":
{"name":"T_1594640403754979b9fb0_c4fd_11ea_9354_f721359f251a_9be04e40-c4fd-11ea-9354-
f721359f251a","description":"","image":"serviceicon_1.svg","category":"CloudServices","bullets":
[{"title":"Cloud Provider","description":"IBM Bluemix
```

```
Platform"}],"providerDisplayName":"IBM","longDescription":"Meant Stack deployed on Multiple VMs to help user
get the stack working is faster
manner","documentationUrl":"http://example.com/meanstack","supportUrl":"http://example.com/meanstack","bindab
le":"","updatable":"","systemTags":true},"tags":[],"actions":
[{"type":"provision","name":"Provision","configuration":{"onError":"pause"},"input_parameters":[],"flow":
{"conditions":[],"templates":[],"resthooks":[{"resthook":
{"title":"resthook_d4961d","id":"resthook7afd540e","headers":{"accept":"application/json","Content-
Type":"application/json"},"url":"https://jsonplaceholder.typicode.com/todos/1","method":"GET"}}],"notificatio
ns":[],"sequence":{"0":"resthook7afd540e"},"error_sequence":{},"abort_sequence":{}},"output_parameters":
[]}],"plans":[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[]}],"version":"1.0.0.0"}},"service_id":"5f0c481b415e1137d7efc5f9","namespaceId":"ibmcom","author":"user2_adm
inistrator","specVersion":"v3","status":"Published","publishedTime":"2020-08-25T14:29:47.231Z","tags":
[{"name":"service_identifier","label":"service_identifier","type":"string","immutable":true,"hidden":false,"r
equired":false,"secured":false,"default":"${svc_instance.service_instance_id}","customtype":"string","permiss
ion":"Read-Only","description":"Unique identifer generated by ICAM which is mapped to the service\n instance
of the template provisioned.","isSystemTag":true},
{"name":"service_name","label":"service_name","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"default":"${svc_instance.name}","customtype":"string","permission":"Read-
Only","description":"Name provided for the service instance by the end user at time of
request","isSystemTag":true},
{"name":"request_user","label":"request_user","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"default":"${svc_instance.owner}","customtype":"string","permission":"Read-
Only","description":"The user id of the current user that requested the provisioning of the
template.","isSystemTag":true},
{"name":"request_group","label":"request_group","type":"string","immutable":true,"hidden":false,"required":fa
lse,"secured":false,"default":"${svc_instance.group}","customtype":"string","permission":"Read-
Only","description":"The current context group id of the current user that requested\n the provisioning of
the template. This is important because a user may be part of\n more than one group and this identifies the
context of which group the user made the
request.","isSystemTag":true}],"systemTags":true,"creationTime":"2020-07-
13T11:40:11.292Z","lastUpdated":"2020-08-
25T14:29:47.231Z","serviceInstanceCount":3,"id":"5f0c481b415e1137d7efc5fa","actions":
[{"type":"provision","name":"Provision","configuration":{"onError":"pause"},"input_parameters":[],"flow":
{"conditions":[],"templates":[],"resthooks":[{"resthook":
{"title":"resthook_d4961d","id":"resthook7afd540e","headers":{"accept":"application/json","Content-
Type":"application/json"},"url":"https://jsonplaceholder.typicode.com/todos/1","method":"GET"}}],"notificatio
ns":[],"sequence":{"0":"resthook7afd540e"},"error_sequence":{},"abort_sequence":{}},"output_parameters":
[]}],"plans":[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[]}],"tagId":"5ecddfe55d6b7600155831de","retiredTime":"2020-08-25T14:29:31.229Z"},
{"version":"2.0.0.0","ServiceEngine_InputFile":{"service":{"specVersion":"v3","catalog_metadata":
{"name":"T_1594640403754979b9fb0_c4fd_11ea_9354_f721359f251a_9be04e40-c4fd-11ea-9354-
f721359f251a","description":"test-with-resthook
desc","image":"serviceicon_1.svg","category":"CloudServices","bullets":
[],"providerDisplayName":"IBM","longDescription":"test-with-resthook long
desc","documentationUrl":"","supportUrl":"","bindable":false,"updatable":"","systemTags":true,"author":"admin
"},"tags":
[{"name":"request_group","label":"request_group","type":"string","immutable":true,"hidden":false,"required":f
alse,"secured":false,"description":"The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and this
identifies the context of which group the user made the
request.","isSystemTag":true,"default":"${svc_instance.group}","customtype":"string","permission":"Read-
Only"},
{"name":"request_user","label":"request_user","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"The user id of the current user that requested the provisioning of the
template.","isSystemTag":true,"default":"${svc_instance.owner}","customtype":"string","permission":"Read-
Only"},
{"name":"service_name","label":"service_name","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"Name provided for the service instance by the end user at time of
request","isSystemTag":true,"default":"${svc_instance.name}","customtype":"string","permission":"Read-Only"},
{"name":"service_identifier","label":"service_identifier","type":"string","immutable":true,"hidden":false,"re
quired":false,"secured":false,"description":"Unique identifer generated by ICAM which is mapped to the
service instance of the template
provisioned.","isSystemTag":true,"default":"${svc_instance.service_instance_id}","customtype":"string","permi
ssion":"Read-Only"}],"actions":[{"type":"provision","name":"Provision","description":"Default action for
deployment of service","input_parameters":
[{"name":"arrayparam","label":"arrayparam","customtype":"list","type":"list","immutable":false,"hidden":false
,"required":false,"secured":false,"default":["first","second","third"],"editable":true}],"flow":
{"conditions":[],"templates":[],"resthooks":[{"resthook":
{"title":"resthook_fb4ca7","id":"resthook1b74da50","headers":{"accept":"application/json","Content-
Type":"application/json"},"url":"http://9.5.28.173:3000/posts","method":"POST","payload":
{"id":2001,"arraylistcustom":"${input_parameters.arrayparam}"}}}],"notifications":[],"sequence":
{"0":"resthook1b74da50"},"error_sequence":{}},"output_parameters":[]}],"plans":
[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[]}],"version":"2.0.0.0"}},"service_id":"5f0c481b415e1137d7efc5f9","tenantId":"72e1331d-3d93-447a-bf61-
de1177f2f0d8","namespaceId":"ibmcom","author":"admin","specVersion":"v3","status":"Published","publishedTime"
:"2020-07-13T11:40:18.421Z","tags":
[{"name":"request_group","label":"request_group","type":"string","immutable":true,"hidden":false,"required":f
alse,"secured":false,"description":"The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and this
identifies the context of which group the user made the
request.","isSystemTag":true,"default":"${svc_instance.group}","customtype":"string","permission":"Read-
```

```
Only"},
{"name":"request_user","label":"request_user","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"The user id of the current user that requested the provisioning of the
template.","isSystemTag":true,"default":"${svc_instance.owner}","customtype":"string","permission":"Read-
Only"},
{"name":"service_name","label":"service_name","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"Name provided for the service instance by the end user at time of
request","isSystemTag":true,"default":"${svc_instance.name}","customtype":"string","permission":"Read-Only"},
{"name":"service_identifier","label":"service_identifier","type":"string","immutable":true,"hidden":false,"re
quired":false,"secured":false,"description":"Unique identifer generated by ICAM which is mapped to the
service instance of the template
provisioned.","isSystemTag":true,"default":"${svc_instance.service_instance_id}","customtype":"string","permi
ssion":"Read-Only"}],"systemTags":true,"creationTime":"2020-07-13T11:40:18.186Z","lastUpdated":"2020-07-
13T11:40:18.421Z","serviceInstanceCount":1,"id":"5f0c4822415e1137d7efc5ff","actions":
[{"type":"provision","name":"Provision","description":"Default action for deployment of
service","input_parameters":
[{"name":"arrayparam","label":"arrayparam","customtype":"list","type":"list","immutable":false,"hidden":false
,"required":false,"secured":false,"default":["first","second","third"],"editable":true}],"flow":
{"conditions":[],"templates":[],"resthooks":[{"resthook":
{"title":"resthook_fb4ca7","id":"resthook1b74da50","headers":{"accept":"application/json","Content-
Type":"application/json"},"url":"http://9.5.28.173:3000/posts","method":"POST","payload":
{"id":2001,"arraylistcustom":"${input_parameters.arrayparam}"}}}],"notifications":[],"sequence":
{"0":"resthook1b74da50"},"error_sequence":{}},"output_parameters":[]}],"plans":
[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[]}],"tagId":"5ecddfe55d6b7600155831de"}]
```


Get Service version manifest by service version ID

This API fetches the service version manifest by using the service version ID.

```
curl -k -X GET \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/${SERVICE_VERSION_ID}?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache'
```

Response:


```
{"version":"1.1.1.1","ServiceEngine_InputFile":{"service":
{"version":"1.1.1.1","specVersion":"v3","catalog_metadata":{"name":"Test_Automationb7550ee0-b6a7-11ea-9f4e-
f3a3b131b5d4","description":"","image":"serviceicon_1.svg","category":"CloudServices","bullets":
[],"providerDisplayName":"IBM","longDescription":"","documentationUrl":"","supportUrl":"","bindable":false,"u
pdatable":"","systemTags":true},"tags":
[{"name":"request_group","label":"request_group","type":"string","immutable":true,"hidden":false,"required":f
alse,"secured":false,"description":"The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and this
identifies the context of which group the user made the
request.","isSystemTag":true,"default":"${svc_instance.group}","customtype":"string","permission":"Read-
Only"},
{"name":"request_user","label":"request_user","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"The user id of the current user that requested the provisioning of the
template.","isSystemTag":true,"default":"${svc_instance.owner}","customtype":"string","permission":"Read-
Only"},
{"name":"service_name","label":"service_name","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"Name provided for the service instance by the end user at time of
request","isSystemTag":true,"default":"${svc_instance.name}","customtype":"string","permission":"Read-Only"},
{"name":"service_identifier","label":"service_identifier","type":"string","immutable":true,"hidden":false,"re
quired":false,"secured":false,"description":"Unique identifer generated by ICAM which is mapped to the
service instance of the template
provisioned.","isSystemTag":true,"default":"${svc_instance.service_instance_id}","customtype":"string","permi
ssion":"Read-Only"}],"actions":[{"type":"provision","name":"Provision","description":"Default action for
deployment of service","input_parameters":[{"name":"url_1","label":"URL
1","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"i
sDynamic":false,"editable":true},{"name":"url_2","label":"URL
2","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"d
escription":"","isDynamic":false,"editable":true}],"flow":{"conditions":[],"templates":[],"resthooks":
[{"resthook":{"title":"resthook_5477d2","id":"resthookee76554d","headers":
{"accept":"application/json","Content-
Type":"application/json"},"immutable":false,"url":"${input_parameters.url_1}","method":"GET"}},{"resthook":
{"title":"resthook_f2fce4","id":"resthook4712f3c3","headers":{"accept":"application/json","Content-
Type":"application/json"},"immutable":false,"url":"${input_parameters.url_2}","method":"GET"}}],"notification
s":[],"sequence":{"0":"resthookee76554d","1":"resthook4712f3c3"},"error_sequence":{}},"output_parameters":
[]}],"plans":[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[]}]}},"service_id":"5ef43b0ccfdfb035e2a3824b","namespaceId":"cert-
manager","author":"user1_administrator","specVersion":"v3","status":"Published","publishedTime":"2020-06-
25T05:50:06.602Z","tags":
[{"name":"request_group","label":"request_group","type":"string","immutable":true,"hidden":false,"required":f
alse,"secured":false,"description":"The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and this
identifies the context of which group the user made the
```

```
request.","isSystemTag":true,"default":"${svc_instance.group}","customtype":"string","permission":"Read-
Only"},
{"name":"request_user","label":"request_user","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"The user id of the current user that requested the provisioning of the
template.","isSystemTag":true,"default":"${svc_instance.owner}","customtype":"string","permission":"Read-
Only"},
{"name":"service_name","label":"service_name","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"Name provided for the service instance by the end user at time of
request","isSystemTag":true,"default":"${svc_instance.name}","customtype":"string","permission":"Read-Only"},
{"name":"service_identifier","label":"service_identifier","type":"string","immutable":true,"hidden":false,"re
quired":false,"secured":false,"description":"Unique identifer generated by ICAM which is mapped to the
service instance of the template
provisioned.","isSystemTag":true,"default":"${svc_instance.service_instance_id}","customtype":"string","permi
ssion":"Read-Only"}],"systemTags":true,"creationTime":"2020-06-25T05:50:04.184Z","lastUpdated":"2020-06-
25T05:50:06.602Z","serviceInstanceCount":0,"id":"5ef43b0ccfdfb035e2a3824c","actions":
[{"type":"provision","name":"Provision","description":"Default action for deployment of
service","input_parameters":[{"name":"url_1","label":"URL
1","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"i
sDynamic":false,"editable":true},{"name":"url_2","label":"URL
2","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"d
escription":"","isDynamic":false,"editable":true}],"flow":{"conditions":[],"templates":[],"resthooks":
[{"resthook":{"title":"resthook_5477d2","id":"resthookee76554d","headers":
{"accept":"application/json","Content-
Type":"application/json"},"immutable":false,"url":"${input_parameters.url_1}","method":"GET"}},{"resthook":
{"title":"resthook_f2fce4","id":"resthook4712f3c3","headers":{"accept":"application/json","Content-
Type":"application/json"},"immutable":false,"url":"${input_parameters.url_2}","method":"GET"}}],"notification
s":[],"sequence":{"0":"resthookee76554d","1":"resthook4712f3c3"},"error_sequence":{}},"output_parameters":
[]}],"plans":[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[]}],"tagId":"5ecddfe55d6b7600155831de","retiredTime":"2020-06-
25T05:50:06.277Z","isParametersResolved":false}
```


Publish a service version

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/${SERVICE_MANIFEST_ID}/publish
?ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache'
```

Response:


```
{"version":"1.0.0.0","ServiceEngine_InputFile":{"service":
{"specVersion":"v3","version":"1.0.0.0","catalog_metadata":{"name":"Test_with_COMPATIBILITY_MODE_ON_16331c60-
ed01-11ea-b709-
43ed3173756c","description":"","image":"serviceicon_1.svg","category":"CloudServices","bullets":
[],"providerDisplayName":"IBM","longDescription":"","documentationUrl":"","supportUrl":"","bindable":false,"u
pdatable":"","systemTags":true},"tags":
[{"name":"request_group","label":"request_group","type":"string","immutable":true,"hidden":false,"required":f
alse,"secured":false,"description":"The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and this
identifies the context of which group the user made the
request.","isSystemTag":true,"default":"${svc_instance.group}","customtype":"string","permission":"Read-
Only"},
{"name":"request_user","label":"request_user","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"The user id of the current user that requested the provisioning of the
template.","isSystemTag":true,"default":"${svc_instance.owner}","customtype":"string","permission":"Read-
Only"},
{"name":"service_name","label":"service_name","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"Name provided for the service instance by the end user at time of
request","isSystemTag":true,"default":"${svc_instance.name}","customtype":"string","permission":"Read-Only"},
{"name":"service_identifier","label":"service_identifier","type":"string","immutable":true,"hidden":false,"re
quired":false,"secured":false,"description":"Unique identifer generated by ICAM which is mapped to the
service instance of the template
provisioned.","isSystemTag":true,"default":"${svc_instance.service_instance_id}","customtype":"string","permi
ssion":"Read-Only"}],"actions":[{"type":"provision","name":"Provision","description":"Default action for
deployment of service","input_parameters":[{"name":"vm-name","label":"vm-
name","customtype":"string","type":"string","immutable":false,"hidden":false,"required":true,"secured":false,
"description":"","isDynamic":false,"default":"vm1","editable":true},
{"name":"Conn1","label":"Conn1","customtype":"connection","type":"connection","immutable":false,"hidden":fals
e,"required":true,"secured":false,"isDynamic":false,"default":"vmware","provider":"VMware
vSphere","editable":true},
{"name":"DataObject1","label":"DataObject1","customtype":"sharedparameter","type":"sharedparameter","immutabl
e":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"Default
NoBastionHostRequired","datatype":"bastionhost","editable":true},
{"name":"region","label":"region","customtype":"string","type":"string","immutable":false,"hidden":false,"req
uired":false,"secured":false,"isDynamic":false,"default":"west1","editable":true},
{"name":"org","label":"Org","customtype":"string","type":"string","immutable":false,"hidden":false,"required"
:false,"secured":false,"description":"","isDynamic":false,"default":"all","editable":true},
{"name":"team","label":"Team","customtype":"string","type":"string","immutable":false,"hidden":false,"require
d":false,"secured":false,"description":"","isDynamic":false,"default":"team1","editable":true},
```

```
{"name":"DataObject2","label":"DataObject2","customtype":"sharedparameter","type":"sharedparameter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"","datatype":"httpproxy","editable":true},
{"name":"aws","label":"aws","customtype":"connection","type":"connection","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"","provider":"Amazon EC2","editable":true},
{"name":"size","label":"size","customtype":"counter","type":"counter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":100,"range":{"min":1,"max":1000},"editable":true}],"flow":{"conditions":[],"templates":[],"resthooks":[{"resthook":{"title":"Prov_R1","id":"resthooka4f9e22b","headers":{"accept":"application/json","Content-Type":"application/json"},"error":false,"immutable":false,"url":"http://9.30.167.199:8080/pingpong/list/1?id=1021&list=${input_parameters.Conn1}&list=${input_parameters.DataObject1}","method":"GET","warning":false}}],"notifications":[],"sequence":{"0":"resthooka4f9e22b"},"error_sequence":{}},"output_parameters":[]}],"plans":[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":[{"name":"DataObject2","label":"DataObject2","customtype":"sharedparameter","type":"sharedparameter","immutable":true,"hidden":true,"required":true,"secured":false,"description":"","isDynamic":false,"default":"DefaultNoProxyRequired","datatype":"httpproxy"},
{"name":"aws","label":"aws","customtype":"connection","type":"connection","immutable":true,"hidden":true,"required":true,"secured":false,"description":"","isDynamic":false,"default":"AWS","provider":"Amazon EC2"},
{"name":"size","label":"size","customtype":"counter","type":"counter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":50,"range":{"min":10,"max":100}},
{"name":"vm-name","label":"vm-name","customtype":"string","type":"string","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"vm1"},
{"name":"Conn1","label":"Conn1","customtype":"connection","type":"connection","immutable":false,"hidden":false,"required":true,"secured":false,"isDynamic":false,"default":"vmware","provider":"VMware vSphere"},
{"name":"DataObject1","label":"DataObject1","customtype":"sharedparameter","type":"sharedparameter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"DefaultNoBastionHostRequired","datatype":"bastionhost"},
{"name":"region","label":"region","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"isDynamic":false,"default":"west1"},
{"name":"org","label":"Org","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"description":"","isDynamic":false,"default":"all"},
{"name":"team","label":"Team","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"description":"","isDynamic":false,"default":"team1"}]},
{"name":"Max","description":"Max","plan_parameters":[{"name":"size","label":"size","customtype":"counter","type":"counter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":100,"range":{"min":100,"max":1000}},
{"name":"vm-name","label":"vm-name","customtype":"string","type":"string","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"vm1"},
{"name":"Conn1","label":"Conn1","customtype":"connection","type":"connection","immutable":false,"hidden":false,"required":true,"secured":false,"isDynamic":false,"default":"vmware","provider":"VMware vSphere"},
{"name":"DataObject1","label":"DataObject1","customtype":"sharedparameter","type":"sharedparameter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"DefaultNoBastionHostRequired","datatype":"bastionhost"},
{"name":"region","label":"region","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"isDynamic":false,"default":"west1"},
{"name":"org","label":"Org","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"description":"","isDynamic":false,"default":"all"},
{"name":"team","label":"Team","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"description":"","isDynamic":false,"default":"team1"},
{"name":"DataObject2","label":"DataObject2","customtype":"sharedparameter","type":"sharedparameter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"","datatype":"httpproxy"},
{"name":"aws","label":"aws","customtype":"connection","type":"connection","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"","provider":"Amazon EC2"}]},
{"name":"Min","description":"Min","plan_parameters":[{"name":"region","label":"region","customtype":"string","type":"string","immutable":true,"hidden":true,"required":false,"secured":false,"isDynamic":false,"default":"west1"},
{"name":"org","label":"Org","customtype":"string","type":"string","immutable":true,"hidden":true,"required":false,"secured":false,"description":"","isDynamic":false,"default":"all"},
{"name":"DataObject2","label":"DataObject2","customtype":"sharedparameter","type":"sharedparameter","immutable":true,"hidden":true,"required":true,"secured":false,"description":"","isDynamic":false,"default":"DefaultNoProxyRequired","datatype":"httpproxy"},
{"name":"aws","label":"aws","customtype":"connection","type":"connection","immutable":true,"hidden":true,"required":true,"secured":false,"description":"","isDynamic":false,"default":"AWS","provider":"Amazon EC2"},
{"name":"size","label":"size","customtype":"counter","type":"counter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":1,"range":{"min":1,"max":10}},
{"name":"vm-name","label":"vm-name","customtype":"string","type":"string","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"vm1"},
{"name":"Conn1","label":"Conn1","customtype":"connection","type":"connection","immutable":false,"hidden":false,"required":true,"secured":false,"isDynamic":false,"default":"vmware","provider":"VMware vSphere"},
{"name":"DataObject1","label":"DataObject1","customtype":"sharedparameter","type":"sharedparameter","immutable":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"DefaultNoBastionHostRequired","datatype":"bastionhost"},
{"name":"team","label":"Team","customtype":"string","type":"string","immutable":false,"hidden":false,"required":false,"secured":false,"description":"","isDynamic":false,"default":"team1"}]}]}},"service_id":"5f4f69cf643bcb0f7b35ab29","tenantId":"72e1331d-3d93-447a-bf61-de1177f2f0d8","namespaceId":"cert-manager","author":"admin","specVersion":"v3","status":"Draft","tags":
```

```
[{"name":"request_group","label":"request_group","type":"string","immutable":true,"hidden":false,"required":f
alse,"secured":false,"description":"The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and this
identifies the context of which group the user made the
request.","isSystemTag":true,"default":"${svc_instance.group}","customtype":"string","permission":"Read-
Only"},
{"name":"request_user","label":"request_user","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"The user id of the current user that requested the provisioning of the
template.","isSystemTag":true,"default":"${svc_instance.owner}","customtype":"string","permission":"Read-
Only"},
{"name":"service_name","label":"service_name","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"description":"Name provided for the service instance by the end user at time of
request","isSystemTag":true,"default":"${svc_instance.name}","customtype":"string","permission":"Read-Only"},
{"name":"service_identifier","label":"service_identifier","type":"string","immutable":true,"hidden":false,"re
quired":false,"secured":false,"description":"Unique identifer generated by ICAM which is mapped to the
service instance of the template
provisioned.","isSystemTag":true,"default":"${svc_instance.service_instance_id}","customtype":"string","permi
ssion":"Read-Only"}],"systemTags":true,"creationTime":"2020-09-
02T09:45:51.304Z","serviceInstanceCount":0,"id":"5f4f69cf643bcb0f7b35ab2a","actions":
[{"type":"provision","name":"Provision","description":"Default action for deployment of
service","input_parameters":[{"name":"vm-name","label":"vm-
name","customtype":"string","type":"string","immutable":false,"hidden":false,"required":true,"secured":false,
"description":"","isDynamic":false,"default":"vm1","editable":true},
{"name":"Conn1","label":"Conn1","customtype":"connection","type":"connection","immutable":false,"hidden":fals
e,"required":true,"secured":false,"isDynamic":false,"default":"vmware","provider":"VMware
vSphere","editable":true},
{"name":"DataObject1","label":"DataObject1","customtype":"sharedparameter","type":"sharedparameter","immutabl
e":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"Default
NoBastionHostRequired","datatype":"bastionhost","editable":true},
{"name":"region","label":"region","customtype":"string","type":"string","immutable":false,"hidden":false,"req
uired":false,"secured":false,"isDynamic":false,"default":"west1","editable":true},
{"name":"org","label":"Org","customtype":"string","type":"string","immutable":false,"hidden":false,"required"
:false,"secured":false,"description":"","isDynamic":false,"default":"all","editable":true},
{"name":"team","label":"Team","customtype":"string","type":"string","immutable":false,"hidden":false,"require
d":false,"secured":false,"description":"","isDynamic":false,"default":"team1","editable":true},
{"name":"DataObject2","label":"DataObject2","customtype":"sharedparameter","type":"sharedparameter","immutabl
e":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"","data
type":"httpproxy","editable":true},
{"name":"aws","label":"aws","customtype":"connection","type":"connection","immutable":false,"hidden":false,"r
equired":true,"secured":false,"description":"","isDynamic":false,"default":"","provider":"Amazon
EC2","editable":true},
{"name":"size","label":"size","customtype":"counter","type":"counter","immutable":false,"hidden":false,"requi
red":true,"secured":false,"description":"","isDynamic":false,"default":100,"range":
{"min":1,"max":1000},"editable":true}],"flow":{"conditions":[],"templates":[],"resthooks":[{"resthook":
{"title":"Prov_R1","id":"resthooka4f9e22b","headers":{"accept":"application/json","Content-
Type":"application/json"},"error":false,"immutable":false,"url":"http://9.30.167.199:8080/pingpong/list/1?
id=1021&list=${input_parameters.Conn1}&list=${input_parameters.DataObject1}","method":"GET","warning":false}}
],"notifications":[],"sequence":{"0":"resthooka4f9e22b"},"error_sequence":{}},"output_parameters":
[]}],"plans":[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[{"name":"DataObject2","label":"DataObject2","customtype":"sharedparameter","type":"sharedparameter","immutab
le":true,"hidden":true,"required":true,"secured":false,"description":"","isDynamic":false,"default":"DefaultN
oProxyRequired","datatype":"httpproxy"},
{"name":"aws","label":"aws","customtype":"connection","type":"connection","immutable":true,"hidden":true,"req
uired":true,"secured":false,"description":"","isDynamic":false,"default":"AWS","provider":"Amazon EC2"},
{"name":"size","label":"size","customtype":"counter","type":"counter","immutable":false,"hidden":false,"requi
red":true,"secured":false,"description":"","isDynamic":false,"default":50,"range":{"min":10,"max":100}},
{"name":"vm-name","label":"vm-
name","customtype":"string","type":"string","immutable":false,"hidden":false,"required":true,"secured":false,
"description":"","isDynamic":false,"default":"vm1"},
{"name":"Conn1","label":"Conn1","customtype":"connection","type":"connection","immutable":false,"hidden":fals
e,"required":true,"secured":false,"isDynamic":false,"default":"vmware","provider":"VMware vSphere"},
{"name":"DataObject1","label":"DataObject1","customtype":"sharedparameter","type":"sharedparameter","immutabl
e":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"Default
NoBastionHostRequired","datatype":"bastionhost"},
{"name":"region","label":"region","customtype":"string","type":"string","immutable":false,"hidden":false,"req
uired":false,"secured":false,"isDynamic":false,"default":"west1"},
{"name":"org","label":"Org","customtype":"string","type":"string","immutable":false,"hidden":false,"required"
:false,"secured":false,"description":"","isDynamic":false,"default":"all"},
{"name":"team","label":"Team","customtype":"string","type":"string","immutable":false,"hidden":false,"require
d":false,"secured":false,"description":"","isDynamic":false,"default":"team1"}]},
{"name":"Max","description":"Max","plan_parameters":
[{"name":"size","label":"size","customtype":"counter","type":"counter","immutable":false,"hidden":false,"requ
ired":true,"secured":false,"description":"","isDynamic":false,"default":100,"range":{"min":100,"max":1000}},
{"name":"vm-name","label":"vm-
name","customtype":"string","type":"string","immutable":false,"hidden":false,"required":true,"secured":false,
"description":"","isDynamic":false,"default":"vm1"},
{"name":"Conn1","label":"Conn1","customtype":"connection","type":"connection","immutable":false,"hidden":fals
e,"required":true,"secured":false,"isDynamic":false,"default":"vmware","provider":"VMware vSphere"},
{"name":"DataObject1","label":"DataObject1","customtype":"sharedparameter","type":"sharedparameter","immutabl
e":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"Default
NoBastionHostRequired","datatype":"bastionhost"},
```

```
{"name":"region","label":"region","customtype":"string","type":"string","immutable":false,"hidden":false,"req
uired":false,"secured":false,"isDynamic":false,"default":"west1"},
{"name":"org","label":"Org","customtype":"string","type":"string","immutable":false,"hidden":false,"required"
:false,"secured":false,"description":"","isDynamic":false,"default":"all"},
{"name":"team","label":"Team","customtype":"string","type":"string","immutable":false,"hidden":false,"require
d":false,"secured":false,"description":"","isDynamic":false,"default":"team1"},
{"name":"DataObject2","label":"DataObject2","customtype":"sharedparameter","type":"sharedparameter","immutabl
e":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"","data
type":"httpproxy"},
{"name":"aws","label":"aws","customtype":"connection","type":"connection","immutable":false,"hidden":false,"r
equired":true,"secured":false,"description":"","isDynamic":false,"default":"","provider":"Amazon EC2"}]},
{"name":"Min","description":"Min","plan_parameters":
[{"name":"region","label":"region","customtype":"string","type":"string","immutable":true,"hidden":true,"requ
ired":false,"secured":false,"isDynamic":false,"default":"west1"},
{"name":"org","label":"Org","customtype":"string","type":"string","immutable":true,"hidden":true,"required":f
alse,"secured":false,"description":"","isDynamic":false,"default":"all"},
{"name":"DataObject2","label":"DataObject2","customtype":"sharedparameter","type":"sharedparameter","immutabl
e":true,"hidden":true,"required":true,"secured":false,"description":"","isDynamic":false,"default":"DefaultNo
ProxyRequired","datatype":"httpproxy"},
{"name":"aws","label":"aws","customtype":"connection","type":"connection","immutable":true,"hidden":true,"req
uired":true,"secured":false,"description":"","isDynamic":false,"default":"AWS","provider":"Amazon EC2"},
{"name":"size","label":"size","customtype":"counter","type":"counter","immutable":false,"hidden":false,"requi
red":true,"secured":false,"description":"","isDynamic":false,"default":1,"range":{"min":1,"max":10}},
{"name":"vm-name","label":"vm-
name","customtype":"string","type":"string","immutable":false,"hidden":false,"required":true,"secured":false,
"description":"","isDynamic":false,"default":"vm1"},
{"name":"Conn1","label":"Conn1","customtype":"connection","type":"connection","immutable":false,"hidden":fals
e,"required":true,"secured":false,"isDynamic":false,"default":"vmware","provider":"VMware vSphere"},
{"name":"DataObject1","label":"DataObject1","customtype":"sharedparameter","type":"sharedparameter","immutabl
e":false,"hidden":false,"required":true,"secured":false,"description":"","isDynamic":false,"default":"Default
NoBastionHostRequired","datatype":"bastionhost"},
{"name":"team","label":"Team","customtype":"string","type":"string","immutable":false,"hidden":false,"require
d":false,"secured":false,"description":"","isDynamic":false,"default":"team1"}]}],"tagId":"5f476f9881758e190d
e1c44e","warning":[]}
```


Duplicate a service

Creates a copy of the Service with all its versions in a new Service.

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICE_ID}/duplicateService?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @service_duplicate.json
```

Payload

```
{
    "service_name": "{{NEW-SERVICE-NAME}}"
}
```

Response


```
 {
  "id": "5df3aacea051bd001d1611cf",
  "name": "SampleV2",
  "author": "admin",
  "image": "serviceicon_1.svg",
  "status": "",
  "serviceManifestId": "5df3aacea051bd001d1611d0",
  "tag": "5c6e73077077fc001715703b",
  "creationTime": "2019-12-13T15:14:22.045Z",
  "lastUpdated": "2019-12-13T15:14:22.156Z",
  "namespaceId": "default",
  "tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
  "description": "A Sample Single Virtual Machine Service",
  "longDescription": "",
  "defaultServiceVersion": {
    "id": "5df3aacea051bd001d1611d0"
    "version": "1.0.0.1",
    "ServiceEngine_InputFile": {
      "service": {
        "version": "1.0.0.1",
        "specVersion": "v3",
        "catalog_metadata": {
          "name": "SampleV2",
          "description": "A Sample Single Virtual Machine Service",
          "image": "serviceicon_1.svg",
          "category": "CloudServices",
```

```
          "bullets": [],
          "providerDisplayName": "IBM",
          "longDescription": "",
          "documentationUrl": "",
          "supportUrl": "",
          "bindable": false,
          "updatable": "",
          "systemTags": true,
          "author": "admin"
        },
        "tags": [
          {
            "name": "request_group",
            "label": "request_group",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The current context group id of the current user that requested the provisioning
of the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
            "isSystemTag": true,
            "default": "${svc_instance.group}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
            "name": "request_user",
            "label": "request_user",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The user id of the current user that requested the provisioning of the
template.",
            "isSystemTag": true,
            "default": "${svc_instance.owner}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
            "name": "service_name",
            "label": "service_name",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Name provided for the service instance by the end user at time of request",
            "isSystemTag": true,
            "default": "${svc_instance.name}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
            "name": "service_identifier",
            "label": "service_identifier",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
            "isSystemTag": true,
            "default": "${svc_instance.service_instance_id}",
            "customtype": "string",
            "permission": "Read-Only"
          }
        ],
        "actions": [
          {
            "type": "provision",
            "name": "Provision",
            "description": "Default action for deployment of service",
            "input_parameters": [
              {
                "name": "vmMemory",
                "label": "vmMemory",
```

```
              "customtype": "string",
              "type": "string",
              "immutable": false,
              "hidden": false,
              "required": false,
              "secured": false
            },
            {
              "name": "vmCpu",
              "label": "vmCpu",
              "customtype": "string",
              "type": "string",
              "immutable": false,
              "hidden": false,
              "required": false,
              "secured": false
            }
          ],
          "flow": {
            "conditions": [],
            "templates": [
              {
                "SingleVirtualMachine": {
                  "title": "SingleVirtualMachine_cca453",
                  "template_name": "SingleVirtualMachine",
                  "version": "2.3",
                  "id": "singlevi23cd3149",
                  "template_type": "Terraform",
                  "template_content_type": "VMware vSphere",
                  "template_provider": "VMware vSphere",
                  "instance_name": "",
                  "cloud_connection_name": "vmware",
                  "template_data_objects": {},
                  "template_params": {
                    "vm_1_name": "ver2vm",
                    "vm_1_dns_servers": [
                      "1.1.1.2"
                    ],
                    "vm_1_dns_suffixes": [
                      "test.cam.com"
                    ],
                    "vm_1_root_disk_keep_on_remove": "false",
                    "vm_1_domain": "test.cam.com",
                    "vm_1-image": "terraform/template/rhel_7_2_small",
                    "vm_1_root_disk_size": "25",
                    "vm_1_root_disk_controller_type": "scsi",
                    "vm_1_root_disk_datastore": "NSXcluster/datastore1",
                    "vm_1_root_disk_type": "eager_zeroed",
                    "vm_1_ipv4_gateway": "10.4.10.1",
                    "vm_1_ipv4_address": "10.4.10.17",
                    "vm_1_memory": "${input_parameters.vmMemory}",
                    "vm_1_ipv4_prefix_length": "23",
                    "vm_1_number_of_vcpu": "${input_parameters.vmCpu}",
                    "vm_1_network_interface_label": "external-net-410",
                    "vm_1_cluster": "icm-vmware-nsx-2",
                    "vm_1_datacenter": "NSX Manage To",
                    "vm_1_folder": "terraform/test",
                    "vm_1_adapter_type": "vmxnet3",
                    "vm_1_resource_pool": "cam"
                  },
                  "outputs": [],
                  "immutable": false,
                  "provider_engine_version": "0.11.11"
                }
              }
            ],
            "resthooks": [],
            "notifications": [
              {
                "emailnotification": {
                  "title": "emailnotification_4278ce",
                  "type": "email",
                  "receiverList": "testmail@test-mail.sample.net",
                  "id": "emailnot21410720",
                  "sender": "CAMadmin",
                  "copyToList": "",
                  "subject": "VM Creation Succeful",
                  "body": "VM create at ${templates.singlevi23cd3149.created_at}",
                  "immutable": false
                }
              },
```

```
                {
                  "emailnotification": {
                    "title": "emailnotification_ac336e",
                    "type": "email",
                    "receiverList": "testmail@test-mail.sample.net",
                    "id": "emailnotd384eeec",
                    "sender": "CAMadmin",
                    "copyToList": "",
                    "subject": "VM Creation failed",
                    "body": "VM creation failure caused by : ${svc_instance.error.message}",
                    "immutable": false
                  }
                }
              ],
              "sequence": {
                "0": "singlevi23cd3149",
                "1": "emailnot21410720"
              },
              "error_sequence": {
                "0": "emailnotd384eeec"
              }
            },
            "output_parameters": []
          }
        ],
        "plans": [
          {
            "name": "Standard",
            "description": "To deploy a Standard plan",
            "actions": [],
            "plan_parameters": []
          }
        ]
      }
    },
    "service_id": "5df3aacea051bd001d1611cf",
    "namespaceId": "default",
    "author": "admin",
    "specVersion": "v3",
    "status": "Draft",
    "tags": [
      {
        "name": "request_group",
        "label": "request_group",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The current context group id of the current user that requested the provisioning of
the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
        "isSystemTag": true,
        "default": "${svc_instance.group}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "request_user",
        "label": "request_user",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The user id of the current user that requested the provisioning of the template.",
        "isSystemTag": true,
        "default": "${svc_instance.owner}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_name",
        "label": "service_name",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Name provided for the service instance by the end user at time of request",
        "isSystemTag": true,
```

```
        "default": "${svc_instance.name}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_identifier",
        "label": "service_identifier",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Unique identifer generated by ICAM which is mapped to the ervice instance of the
template provisioned.",
        "isSystemTag": true,
        "default": "${svc_instance.service_instance_id}",
        "customtype": "string",
        "permission": "Read-Only"
      }
    ],
    "systemTags": true,
    "creationTime": "2019-12-13T15:14:22.062Z",
    "serviceInstanceCount": 0,
  },
  "warning": []
}
```

Copy service version to a service

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/${SOURCE_SERVICE_VERSION_ID}/copyService?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @service_version_copy.json
```

Payload

```
{
  "to_service_id": "{{SERVICE_ID}}",
  "new_version_name": "{{NEW-VERSION-LABEL}}"
}
```

Response

```
{
  "id": "5df3bb2fa051bd001d1611e1",
  "version": "1.0.0.2",
  "ServiceEngine_InputFile": {
    "service": {
      "version": "1.0.0.2",
      "specVersion": "v3",
      "catalog_metadata": {
        "name": "SampleV2",
        "description": "A Sample Single Virtual Machine Service",
        "image": "serviceicon_1.svg",
        "category": "CloudServices",
        "bullets": [],
        "providerDisplayName": "IBM",
        "longDescription": "",
        "documentationUrl": "",
        "supportUrl": "",
        "bindable": false,
        "updatable": "",
        "systemTags": true,
        "author": "admin"
      },
      "tags": [
        {
          "name": "request_group",
          "label": "request_group",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "The current context group id of the current user that requested the provisioning of
the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
          "isSystemTag": true,
```

```
          "default": "${svc_instance.group}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
          "name": "request_user",
          "label": "request_user",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "The user id of the current user that requested the provisioning of the template.",
          "isSystemTag": true,
          "default": "${svc_instance.owner}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
          "name": "service_name",
          "label": "service_name",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "Name provided for the service instance by the end user at time of request",
          "isSystemTag": true,
          "default": "${svc_instance.name}",
          "customtype": "string",
          "permission": "Read-Only"
        },
        {
          "name": "service_identifier",
          "label": "service_identifier",
          "type": "string",
          "immutable": true,
          "hidden": false,
          "required": false,
          "secured": false,
          "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
          "isSystemTag": true,
          "default": "${svc_instance.service_instance_id}",
          "customtype": "string",
          "permission": "Read-Only"
        }
      ],
      "actions": [
        {
          "type": "provision",
          "name": "Provision",
          "description": "Default action for deployment of service",
          "input_parameters": [
            {
              "name": "vmMemory",
              "label": "vmMemory",
              "customtype": "string",
              "type": "string",
              "immutable": false,
              "hidden": false,
              "required": false,
              "secured": false
            },
            {
              "name": "vmCpu",
              "label": "vmCpu",
              "customtype": "string",
              "type": "string",
              "immutable": false,
              "hidden": false,
              "required": false,
              "secured": false
            }
          ],
          "flow": {
            "conditions": [],
            "templates": [
              {
                "SingleVirtualMachine": {
                  "title": "SingleVirtualMachine_cca453",
```

```
                "template_name": "SingleVirtualMachine",
                "version": "2.3",
                "id": "singlevi23cd3149",
                "template_type": "Terraform",
                "template_content_type": "VMware vSphere",
                "template_provider": "VMware vSphere",
                "instance_name": "",
                "cloud_connection_name": "vmware",
                "template_data_objects": {},
                "template_params": {
                  "vm_1_name": "ver2vm",
                  "vm_1_dns_servers": [
                    "1.1.1.2"
                  ],
                  "vm_1_dns_suffixes": [
                    "test.cam.com"
                  ],
                  "vm_1_root_disk_keep_on_remove": "false",
                  "vm_1_domain": "test.cam.com",
                  "vm_1-image": "terraform/template/rhel_7_2_small",
                  "vm_1_root_disk_size": "25",
                  "vm_1_root_disk_controller_type": "scsi",
                  "vm_1_root_disk_datastore": "NSXcluster/datastore1",
                  "vm_1_root_disk_type": "eager_zeroed",
                  "vm_1_ipv4_gateway": "10.4.10.1",
                  "vm_1_ipv4_address": "10.4.10.17",
                  "vm_1_memory": "${input_parameters.vmMemory}",
                  "vm_1_ipv4_prefix_length": "23",
                  "vm_1_number_of_vcpu": "${input_parameters.vmCpu}",
                  "vm_1_network_interface_label": "external-net-410",
                  "vm_1_cluster": "icm-vmware-nsx-2",
                  "vm_1_datacenter": "NSX Manage To",
                  "vm_1_folder": "terraform/test",
                  "vm_1_adapter_type": "vmxnet3",
                  "vm_1_resource_pool": "cam"
                },
                "outputs": [],
                "immutable": false
              }
            }
          ],
          "resthooks": [],
          "notifications": [
            {
              "emailnotification": {
                "title": "emailnotification_4278ce",
                "type": "email",
                "receiverList": "testmail@testr-mail.net",
                "id": "emailnot21410720",
                "sender": "CAMadmin",
                "copyToList": "",
                "subject": "VM Creation Succeful",
                "body": "VM create at ${templates.singlevi23cd3149.created_at}",
                "immutable": false
              }
            }
          ],
          "sequence": {
            "0": "singlevi23cd3149",
            "1": "emailnot21410720"
          },
          "error_sequence": {}
        },
        "output_parameters": []
      }
    ],
    "plans": [
      {
        "name": "Standard",
        "description": "To deploy a Standard plan",
        "actions": [],
        "plan_parameters": []
      }
    ]
  }
},
"service_id": "5df3b815a051bd001d1611da",
"tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
"namespaceId": "default",
"author": "admin",
"specVersion": "v3",
```

```
  "status": "Draft",
  "tags": [
    {
      "name": "request_group",
      "label": "request_group",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "The current context group id of the current user that requested the provisioning of the
template. This is important because a user may be part of more than one group and this identifies the context
of which group the user made the request.",
      "isSystemTag": true,
      "default": "${svc_instance.group}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "request_user",
      "label": "request_user",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "The user id of the current user that requested the provisioning of the template.",
      "isSystemTag": true,
      "default": "${svc_instance.owner}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "service_name",
      "label": "service_name",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "Name provided for the service instance by the end user at time of request",
      "isSystemTag": true,
      "default": "${svc_instance.name}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "service_identifier",
      "label": "service_identifier",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
      "isSystemTag": true,
      "default": "${svc_instance.service_instance_id}",
      "customtype": "string",
      "permission": "Read-Only"
    }
  ],
  "systemTags": true,
  "creationTime": "2019-12-13T16:24:15.334Z",
  "serviceInstanceCount": 0,
  "warning": []
}
```

Change the category of a service

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICE_ID}/updateServiceDetails?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
 -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
 -H 'Content-Type: application/json' \
 -H 'cache-control: no-cache' \
 -d @service_change_category.json
```

Payload, content for service_change_category.json

```
{
  "service": {
    "catalog_metadata": {
      "category": "{{NAME-OF-CATEGORY}}"
    }
  }
}
```

Response

```
{
  "service": {
    "name": "SampleV1",
    "author": "admin",
    "image": "serviceicon_1.svg",
    "status": "",
    "serviceManifestId": "5df3aacea051bd001d1611d0",
    "tag": "5d8a3531bbcac2001d967d1d",
    "creationTime": "2019-12-13T15:14:22.045Z",
    "lastUpdated": "2019-12-13T15:30:09.054Z",
    "namespaceId": "default",
    "tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
    "id": "5df3aacea051bd001d1611cf",
    "description": "A Sample Single Virtual Machine Service",
    "longDescription": ""
  }
}
```

Rename a service

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICE_ID}/updateServiceDetails?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @service_rename.json
```

Payload, content for service_rename.json

```
{
    "service": {
        "catalog_metadata": {
            "name": "SampleV3"
        }
    }
}
```

Response

```
{
  "service": {
    "id": "5df3aacea051bd001d1611cf",
    "name": "SampleV3",
    "author": "admin",
    "image": "serviceicon_1.svg",
    "serviceManifestId": "5df3aacea051bd001d1611d0",
    "tag": "5c6e73077077fc001715703b",
    "creationTime": "2019-12-13T15:14:22.045Z",
    "lastUpdated": "2019-12-13T15:22:38.185Z",
    "namespaceId": "default",
    "tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
    "description": "A Sample Single Virtual Machine Service",
    "longDescription": ""
  }
}
```

Change service image for a service

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/${SERVICE_ID}/updateServiceDetails?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
 -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
 -H 'Content-Type: application/json' \
 -H 'cache-control: no-cache' \
 -d @service_change_image.json
```

Payload: content for service_change_image.json

```
{
  "service": {
```

```
    "catalog_metadata": {
      "image": "{{NEW-IMAGE-NAME}}"
    }
  }
}
```

Response

```
{
  "service": {
    "name": "SampleV1",
    "author": "admin",
    "image": "serviceicon_2.svg",
    "status": ""
    "serviceManifestId": "5df3aacea051bd001d1611d0",
    "tag": "5d8a3531bbcac2001d967d1d",
    "creationTime": "2019-12-13T15:14:22.045Z",
    "lastUpdated": "2019-12-13T15:30:09.054Z",
    "namespaceId": "default",
    "tenantId": "674a6a42-76c0-450c-a225-e743772c49f3",
    "id": "5df3aacea051bd001d1611cf",
    "description": "A Sample Single Virtual Machine Service",
    "longDescription": ""
  }
}
```

Assign access to a service

Assigns a single or multiple namespaces to a service.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Accept: application/json" "https://${TSA_HOSTNAME}/cam/composer/api/v1/Service/<id>/setNamespaceId?
tenantId=<tenantId>&cloudOE_spaceGuid=<comma separated list of namesapces>"
```

For scenarios of assigning access, see Assigning access to a service

Retire a service version

This API retires both published services and draft services.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/${SERVICE_VERSION_ID}/retire?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}
```

Unretire a service version

This API unretires the already retired service and moves it back to Published state.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/${SERVICE_VERSION_ID}/unretire?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}
```

Auto-retire a service version

This API retires a service version on the set date. The service version is retired at mid-night of the set day.

```
curl -k --location --request POST
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/${SERVICE_VERSION_ID}/setRetireOnDate?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}' \
-H 'Accept: application/json' \
-H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
-H 'Content-Type: application/json' \
-d @set_retire_on_date.json
```

set_retire_on_date.json:

```
{
    "retire_on_date": "2020-10-28T07:03:11.612Z"
}
```

Service Instance APIs

Deploy a service instance

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/{SERVICEMANIFEST_ID}/deploy?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @service__instance_create.json
```

service__instance_create.json:

```
{
  "name": "Service_Instance_Name",
  "instance_plan": "Standard",
  "instance_plan_id": "5dc13f03ec19fe001508e539",
  "tags_input": [
    {
      "value": "${svc_instance.group}",
      "label": "request_group",
      "name": "request_group",
      "secured": false
    },
    {
      "value": "${svc_instance.owner}",
      "label": "request_user",
      "name": "request_user",
      "secured": false
    },
    {
      "value": "${svc_instance.name}",
      "label": "service_name",
      "name": "service_name",
      "secured": false
    },
    {
      "value": "${svc_instance.service_instance_id}",
      "label": "service_identifier",
      "name": "service_identifier",
      "secured": false
    }
  ],
  "action": "Provision",
  "resourceProperties": {
    "templateParameters": [],
    "specifications": []
  },
  "ownership": {
    "username": "",
    "displayName": "admin",
    "orgGuid": "all",
    "spaceGuid": "abcdef123456",
    "tenantId": ""
  }
}
```

Response:


```
{"name":
"Service_Instance_Name","Status":"Submitted","owner":"admin","templateName":"","ordered_time":"2020-08-
25T06:15:42.245Z","updated_time":"2020-08-25T06:15:42.246Z","creationTime":"2020-08-
25T06:15:42.245Z","activity_deployment_details":{},"logs":"Waiting for service action to
start","serviceId":"5f0c481b415e1137d7efc5f9","serviceManifestId":"5f0c481b415e1137d7efc5fa","ownership":
{"username":"admin","displayName":"admin","orgGuid":"all","spaceGuid":"ibmcom","tenantId":"4c46c3f2-6582-
49de-b697-4891ae7e8dee"},"namespaceId":"ibmcom","tenantId":"4c46c3f2-6582-49de-b697-
4891ae7e8dee","instance_plan":"Standard","action":"Provision","id":"5f44ac8ee32082001cec758a","ServiceID":"5f
0c481b415e1137d7efc5f9","instance_plan_id":"5f0c481b415e1137d7efc5fc","resourceProperties":
{"templateParameters":[],"specifications":[]},"catalogItemId":"5f0c481b415e1137d7efc5f9","tags_input":
[{"value":"5f44ac8ee32082001cec758a","label":"service_identifier","name":"service_identifier","secured":false
},{"value":"s9","label":"service_name","name":"service_name","secured":false},
{"value":"admin","label":"request_user","name":"request_user","secured":false},
{"value":"admin","label":"request_group","name":"request_group","secured":false}],"ace_orgGuid":"all","cloudO
E_spaceGuid":"ibmcom","Bearer_Token":"*******","bindable":false,"catalogItemName":"T_1594640403754979b9fb0_c4
fd_11ea_9354_f721359f251a_9be04e40-c4fd-11ea-9354-f721359f251a","serviceManifest":
{"version":"1.0.0.0","ServiceEngine_InputFile":{"service":{"specVersion":"v3","catalog_metadata":
{"name":"T_1594640403754979b9fb0_c4fd_11ea_9354_f721359f251a_9be04e40-c4fd-11ea-9354-
f721359f251a","description":"","image":"serviceicon_1.svg","category":"CloudServices","bullets":
[{"title":"Cloud Provider","description":"IBM Bluemix
Platform"}],"providerDisplayName":"IBM","longDescription":"Meant Stack deployed on Multiple VMs to help user
get the stack working is faster
manner","documentationUrl":"http://example.com/meanstack","supportUrl":"http://example.com/meanstack","bindab
```

```
le":"","updatable":"","systemTags":true},"tags":[],"actions":
[{"type":"provision","name":"Provision","configuration":{"onError":"pause"},"input_parameters":[],"flow":
{"conditions":[],"templates":[],"resthooks":[{"resthook":
{"title":"resthook_d4961d","id":"resthook7afd540e","headers":{"accept":"application/json","Content-
Type":"application/json"},"url":"https://jsonplaceholder.typicode.com/todos/1","method":"GET"}}],"notificatio
ns":[],"sequence":{"0":"resthook7afd540e"},"error_sequence":{},"abort_sequence":{}},"output_parameters":
[]}],"plans":[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[]}],"version":"1.0.0.0"}},"service_id":"5f0c481b415e1137d7efc5f9","namespaceId":"ibmcom","author":"user2_adm
inistrator","specVersion":"v3","status":"Published","publishedTime":"2020-07-13T11:40:17.901Z","tags":
[{"name":"service_identifier","label":"service_identifier","type":"string","immutable":true,"hidden":false,"r
equired":false,"secured":false,"default":"${svc_instance.service_instance_id}","customtype":"string","permiss
ion":"Read-Only","description":"Unique identifer generated by ICAM which is mapped to the service\n instance
of the template provisioned.","isSystemTag":true},
{"name":"service_name","label":"service_name","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"default":"${svc_instance.name}","customtype":"string","permission":"Read-
Only","description":"Name provided for the service instance by the end user at time of
request","isSystemTag":true},
{"name":"request_user","label":"request_user","type":"string","immutable":true,"hidden":false,"required":fals
e,"secured":false,"default":"${svc_instance.owner}","customtype":"string","permission":"Read-
Only","description":"The user id of the current user that requested the provisioning of the
template.","isSystemTag":true},
{"name":"request_group","label":"request_group","type":"string","immutable":true,"hidden":false,"required":fa
lse,"secured":false,"default":"${svc_instance.group}","customtype":"string","permission":"Read-
Only","description":"The current context group id of the current user that requested\n the provisioning of
the template. This is important because a user may be part of\n more than one group and this identifies the
context of which group the user made the
request.","isSystemTag":true}],"systemTags":true,"creationTime":"2020-07-
13T11:40:11.292Z","lastUpdated":"2020-07-
13T11:40:17.901Z","serviceInstanceCount":2,"id":"5f0c481b415e1137d7efc5fa","actions":
[{"type":"provision","name":"Provision","configuration":{"onError":"pause"},"input_parameters":[],"flow":
{"conditions":[],"templates":[],"resthooks":[{"resthook":
{"title":"resthook_d4961d","id":"resthook7afd540e","headers":{"accept":"application/json","Content-
Type":"application/json"},"url":"https://jsonplaceholder.typicode.com/todos/1","method":"GET"}}],"notificatio
ns":[],"sequence":{"0":"resthook7afd540e"},"error_sequence":{},"abort_sequence":{}},"output_parameters":
[]}],"plans":[{"name":"Standard","description":"To deploy a Standard plan","actions":[],"plan_parameters":
[]}],"tagId":"5ecddfe55d6b7600155831de"},"action_status":{},"output_parameters":[],"bind_array":{}}
```


Get the list of deployed instance: GET call

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" "https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Where **namespace** is **namespace Id** and **team** is **organization Id**.

Headers required:

- Content-Type: **application/json**
- Authorization: **<bearer token>**

Get details of a Service Instance by using the id

This GET call fetches the details of a specific by its id:

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}"
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/<service-instance-id>?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Headers remain the same as in the terminate service instance call.

Get Input Parameters of a service instance

The API retrieves the inputs parameters of a service instance. It is recommended to use this API before Retry/Resume/Edit service instance so that you can verify the input parameters before Retry/Resume/Edit service instance. This API also resolves the shared parameters and parameters of type connection, if any. Earlier this API (**GET ServiceManifest/5e4f5a51826bc70015dfaece?service_filter[resolve_params][action_name]=Provision**) was used for the same purpose.

Note: It is recommended to use the **getInputParameters** API instead of **ServiceManifest** API if you want to try Retry or Resume after a service version change.

To use this API, you need a service instance ID. To get the service instance ID:

1. Retrieve all the service instances using the following API Get the list of deployed instance. The response contains an array of service instances
2. Locate the instance by name and get the ID.
3. Use the following curl command to get the input parameters for the service instance:

```
curl --location --request GET \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/${ServiceInstanceId}/getInputParameters?
cloudOE_spaceGuid=development&ace_orgGuid=all&tenantId=c26b3dd4-4e74-4545-8ac9-bc30a6167b33' \
--header 'Content-Type: application/json'
```

Response


```
{
  "name": "switch_resthookIns",
  "Status": "Error",
  "owner": "admin",
  "active_time": "2020-02-27T05:14:10+00:00",
  "templateName": "",
  "ordered_time": "2020-02-27T05:13:06.943Z",
  "updated_time": "2020-02-27T05:13:06.943Z",
  "creationTime": "2020-02-27T05:13:06.943Z",
  "instance_id": "d7395b20-591f-11ea-bab8-5f5432f25af5",
  "activity_deployment_details": {
    "instance": {
      "vmwarens666346e3": {
        "action": "CREATE",
        "activity_id": "vmwarens666346e3",
        "connection_name": "NSX",
        "created_at": "2020-02-27T05:13:14.914Z",
        "fail_quietly": "false",
        "flow_control_#": "1",
        "id": "5e574feaf63b0a001564e6db",
        "instance_name": "switch_resthookIns-00-inst",
        "long_running_delay": "0",
        "long_running_timeout": "0",
        "provider_engine_version": "0.11.11",
        "resource_id": "serviceblueprint_stack.vmwarens666346e3",
        "sequence_no": "1",
        "stack_data_%": "1",
        "stack_data_latest_state_id": "5e574febf63b0a001564e6dc",
        "stack_error_message": "",
        "stack_id": "5e574feaf63b0a001564e6db",
        "stack_url": "",
        "status": "SUCCESS",
        "status_error_message": "",
        "status_message": "[SUCCESS] ",
        "template_name": "VMware NSX-T Sample to create a logical switch",
        "template_outputs_%": "0",
        "template_params_%": "7",
        "template_params_nsxt_logical_switch_desc": "des",
        "template_params_nsxt_logical_switch_name": "name",
        "template_params_nsxt_logical_switch_repl_mode": "MTEP",
        "template_params_nsxt_logical_switch_state": "UP",
        "template_params_nsxt_tag": "Auto",
        "template_params_nsxt_tag_scope": "Auto-Project",
        "template_params_nsxt_transport_zone": "CAMDC2Zone",
        "template_provider": "VMware NSX-T",
        "template_tags_%": "4",
        "template_tags_request_group": "admin",
        "template_tags_request_user": "admin",
        "template_tags_service_identifier": "5e574fe2dc44840015c8093b",
        "template_tags_service_name": "switch_resthookins",
        "template_type": "TERRAFORM",
        "template_version": "2.3",
        "service_action_type": "provision",
        "service_action_name": "Provision",
        "action_instance_name": "switch_resthookIns",
        "state": "CREATED",
        "flow_control_0.case": "",
        "flow_control_0.case_is_selected": "",
        "flow_control_0.continue_failed": "false",
        "flow_control_0.continue_on_prev_failed": "false",
        "flow_control_0.continue_on_prev_inprogress": "false",
        "flow_control_0.depends_on.#": "1",
        "flow_control_0.depends_on.0": "serviceblueprint_action_start.",
        "flow_control_0.meta.%": "0",
        "flow_control_0.prev_resource_sequence_no": "0",
        "flow_control_0.prev_resource_status": "SUCCESS",
        "flow_control_0.resume": "false",
        "flow_control_0.resume_version": "0",
        "flow_control_0.version": "0"
      }
    },
    "output_parameters": {}
  },
  "logs": "serviceblueprint_action_start.start: Refreshing state... (ID:
```

```
provision)\nserviceblueprint_stack.vmwarens666346e3: Refreshing state... (ID:
5e574feaf63b0a001564e6db)\nserviceblueprint_http.resthook2ec4d4f7: Refreshing state... (ID:
resthook2ec4d4f7)\nserviceblueprint_http.resthook16dc50cc: Refreshing state... (ID:
resthook16dc50cc)\nserviceblueprint_http.resthook16dc50cc: Destroying... (ID:
resthook16dc50cc)\nserviceblueprint_http.resthook2ec4d4f7: Modifying... (ID: resthook2ec4d4f7)\n  url:
\"https://jsonplaceholder.typicode.com/todos/1\" =>
\"https://jsonplaceholder.Invalid\"\nserviceblueprint_http.resthook2ec4d4f7: Modifications complete
after 0s (ID: resthook2ec4d4f7)\nserviceblueprint_http.resthook16dc50cc: Still destroying... (ID:
resthook16dc50cc, 10s elapsed)\nserviceblueprint_http.resthook16dc50cc: Destruction complete after
11s\n\nApply complete! Resources: 0 added, 1 changed, 1
destroyed.\n\nOutputs:\n\nresthook2ec4d4f7_message = { \"resthook_88f097\": { \"status\": \"FAILED\" ,
\"code\": \"502\" ,  \"message\": \"Failure caused by : <nil>\" }}\nresthook2ec4d4f7_status =
FAILED\nresthook2ec4d4f7_status_error_message = <nil>\nvmwarens666346e3_message = {
\"switch_resthookIns-00-inst\": { \"status\": \"SUCCESS\" ,  \"action\": \"CREATE\"
}}\nvmwarens666346e3_status = SUCCESS\nvmwarens666346e3_status_create_error_message =
\nvmwarens666346e3_status_error_message =\n",
  "serviceId": "5e4f59ca826bc70015dfaec8",
  "serviceManifestId": "5e4f5a51826bc70015dfaece",
  "ownership": {
    "username": "admin",
    "displayName": "admin",
    "orgGuid": "all",
    "spaceGuid": "development",
    "tenantId": "c26b3dd4-4e74-4545-8ac9-bc30a6167b33"
  },
  "namespaceId": "development",
  "tenantId": "c26b3dd4-4e74-4545-8ac9-bc30a6167b33",
  "instance_plan": "Standard",
  "action": "Provision",
  "id": "5e574fe2dc44840015c8093b",
  "ServiceID": "5e4f59ca826bc70015dfaec8",
  "instance_plan_id": "5e57566fdc44840015c8094c",
  "resourceProperties": {
    "templateParameters": [],
      "specifications": [
      {
        "label": "Tansport Zone Name",
        "name": "nsxt_transport_zone",
        "value": "CAMDC2Zone",
        "secured": false
      },
      {
        "label": "Rest URL",
        "name": "url",
        "value": "https://jsonplaceholder.Invalid",
        "secured": false
      }
    ]
  },
  "catalogItemId": "5e4f59ca826bc70015dfaec8",
  "tags_input": [
    {
      "value": "admin",
      "label": "request_group",
      "name": "request_group",
      "secured": false
    },
    {
      "value": "admin",
      "label": "request_user",
      "name": "request_user",
      "secured": false
    },
    {
      "value": "switch_resthookins",
      "label": "service_name",
      "name": "service_name",
      "secured": false
    },
    {
      "value": "5e574fe2dc44840015c8093b",
      "label": "service_identifier",
      "name": "service_identifier",
      "secured": false
    }
  ],
  "ace_orgGuid": "all",
  "cloudOE_spaceGuid": "development",
  "Bearer_Token": "*******",
  "bindable": false,
  "catalogItemName": "switch_resthook",
```

```
  "serviceManifest": {
    "version": "2.2.2.2",
    "ServiceEngine_InputFile": {
      "service": {
        "version": "2.2.2.2",
        "specVersion": "v3",
        "catalog_metadata": {
          "name": "switch_resthook",
          "description": "",
          "image": "serviceicon_1.svg",
          "category": "CloudServices",
          "bullets": [],
          "providerDisplayName": "IBM",
          "longDescription": "",
          "documentationUrl": "",
          "supportUrl": "",
          "bindable": false,
          "updatable": "",
          "systemTags": true
        },
        "tags": [
          {
            "name": "request_group",
            "label": "request_group",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and
this identifies the context of which group the user made the request.",
            "isSystemTag": true,
            "default": "${svc_instance.group}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
            "name": "request_user",
            "label": "request_user",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The user id of the current user that requested the provisioning of the
template.",
            "isSystemTag": true,
            "default": "${svc_instance.owner}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
            "name": "service_name",
            "label": "service_name",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Name provided for the service instance by the end user at time of request",
            "isSystemTag": true,
            "default": "${svc_instance.name}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
            "name": "service_identifier",
            "label": "service_identifier",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Unique identifer generated by ICAM which is mapped to the service instance
of the template provisioned.",
            "isSystemTag": true,
            "default": "${svc_instance.service_instance_id}",
            "customtype": "string",
            "permission": "Read-Only"
          }
```

```
        ],
        "upgrade_from": [
          {
            "version": "1.1.1.1"
          }
        ],
        "downgrade_to": [
          {
            "version": "1.1.1.1"
          }
        ],
        "actions": [
          {
            "type": "provision",
            "name": "Provision",
            "description": "Default action for deployment of service",
            "input_parameters": [
              {
                "name": "nsxt_transport_zone",
                "label": "Tansport Zone Name",
                "customtype": "string",
                "type": "string",
                "immutable": false,
                "hidden": false,
                "required": false,
                "secured": false,
                "description": "The name of the Transport Zone to be used by the logical switch.",
                "default": "CAMDC2Zone",
                "editable": true
              },
              {
                "name": "url",
                "label": "Rest URL",
                "customtype": "string",
                "type": "string",
                "immutable": false,
                "hidden": false,
                "required": false,
                "secured": false,
                "isDynamic": false,
                "editable": true,
                "default": "https://jsonplaceholder.typicode.com/todos/1"
              }
            ],
            "flow": {
              "conditions": [],
              "templates": [
                {
                  "VMware NSX-T Sample to create a logical switch": {
                    "title": "VMware NSX-T Sample to create a logical switch_b70a6e",
                    "template_name": "VMware NSX-T Sample to create a logical switch",
                    "version": "2.3",
                    "id": "vmwarens666346e3",
                    "template_type": "Terraform",
                    "template_content_type": "VMware NSX-T",
                    "template_provider": "VMware NSX-T",
                    "instance_name": "",
                    "cloud_connection_name": "NSX",
                    "template_data_objects": {},
                    "template_params": {
                      "nsxt_tag_scope": "Auto-Project",
                      "nsxt_tag": "Auto",
                      "nsxt_transport_zone": "CAMDC2Zone",
                      "nsxt_logical_switch_name": "name",
                      "nsxt_logical_switch_desc": "des",
                      "nsxt_logical_switch_state": "UP",
                      "nsxt_logical_switch_repl_mode": "MTEP"
                    },
                    "outputs": [],
                    "editable": true,
                    "provider_engine_version": "0.11.11"
                  }
                }
              ],
              "resthooks": [
                {
                  "resthook": {
                    "title": "resthook_88f097",
                    "id": "resthook2ec4d4f7",
                    "headers": {
                      "accept": "application/json",
```

```
                    "Content-Type": "application/json"
                  },
                  "editable": true,
                  "url": "${input_parameters.url}",
                  "method": "GET"
                }
              }
            ],
            "notifications": [],
            "sequence": {
              "0": "vmwarens666346e3",
              "1": "resthook2ec4d4f7"
            },
            "error_sequence": {}
          },
          "output_parameters": []
        }
      ],
      "plans": [
        {
          "name": "Standard",
          "description": "To deploy a Standard plan",
          "actions": [],
          "plan_parameters": []
        }
      ]
    }
  },
  "service_id": "5e4f59ca826bc70015dfaec8",
  "tenantId": "c26b3dd4-4e74-4545-8ac9-bc30a6167b33",
  "namespaceId": "",
  "author": "admin",
  "specVersion": "v3",
  "status": "Draft",
  "tags": [
    {
      "name": "request_group",
      "label": "request_group",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "The current context group id of the current user that requested the provisioning
of the template. This is important because a user may be part of more than one group and this identifies
the context of which group the user made the request.",
      "isSystemTag": true,
      "default": "${svc_instance.group}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "request_user",
      "label": "request_user",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "The user id of the current user that requested the provisioning of the
template.",
      "isSystemTag": true,
      "default": "${svc_instance.owner}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
      "name": "service_name",
      "label": "service_name",
      "type": "string",
      "immutable": true,
      "hidden": false,
      "required": false,
      "secured": false,
      "description": "Name provided for the service instance by the end user at time of request",
      "isSystemTag": true,
      "default": "${svc_instance.name}",
      "customtype": "string",
      "permission": "Read-Only"
    },
    {
```

```
        "name": "service_identifier",
        "label": "service_identifier",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Unique identifer generated by ICAM which is mapped to the service instance of
the template provisioned.",
        "isSystemTag": true,
        "default": "${svc_instance.service_instance_id}",
        "customtype": "string",
        "permission": "Read-Only"
      }
    ],
    "systemTags": true,
    "creationTime": "2020-02-21T04:19:29.736Z",
    "serviceInstanceCount": 5,
    "id": "5e4f5a51826bc70015dfaece",
    "actions": [
      {
        "type": "provision",
        "name": "Provision",
        "description": "Default action for deployment of service",
        "input_parameters": [
          {
            "name": "nsxt_transport_zone",
            "label": "Tansport Zone Name",
            "customtype": "string",
            "type": "string",
            "immutable": false,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The name of the Transport Zone to be used by the logical switch.",
            "default": "CAMDC2Zone",
            "editable": true
          },
          {
            "name": "url",
            "label": "Rest URL",
            "customtype": "string",
            "type": "string",
            "immutable": false,
            "hidden": false,
            "required": false,
            "secured": false,
            "isDynamic": false,
            "editable": true,
            "default": "https://jsonplaceholder.typicode.com/todos/1"
          }
        ],
        "flow": {
          "conditions": [],
          "templates": [
            {
              "VMware NSX-T Sample to create a logical switch": {
                "title": "VMware NSX-T Sample to create a logical switch_b70a6e",
                "template_name": "VMware NSX-T Sample to create a logical switch",
                "version": "2.3",
                "id": "vmwarens666346e3",
                "template_type": "Terraform",
                "template_content_type": "VMware NSX-T",
                "template_provider": "VMware NSX-T",
                "instance_name": "",
                "cloud_connection_name": "NSX",
                "template_data_objects": {},
                "template_params": {
                  "nsxt_tag_scope": "Auto-Project",
                  "nsxt_tag": "Auto",
                  "nsxt_transport_zone": "CAMDC2Zone",
                  "nsxt_logical_switch_name": "name",
                  "nsxt_logical_switch_desc": "des",
                  "nsxt_logical_switch_state": "UP",
                  "nsxt_logical_switch_repl_mode": "MTEP"
                },
                "outputs": [],
                "editable": true,
                "provider_engine_version": "0.11.11"
              }
            }
```

```
          ],
          "resthooks": [
            {
              "resthook": {
                "title": "resthook_88f097",
                "id": "resthook2ec4d4f7",
                "headers": {
                  "accept": "application/json",
                  "Content-Type": "application/json"
                },
                "editable": true,
                "url": "${input_parameters.url}",
                "method": "GET"
              }
            }
          ],
          "notifications": [],
          "sequence": {
            "0": "vmwarens666346e3",
            "1": "resthook2ec4d4f7"
          },
          "error_sequence": {}
        },
        "output_parameters": []
      }
    ],
    "plans": [
      {
        "name": "Standard",
        "description": "To deploy a Standard plan",
        "actions": [],
        "plan_parameters": []
      }
    ],
    "tagId": "5e4f59ca826bc70015dfaec7"
  },
  "action_status": {},
  "output_parameters": [],
  "bind_array": {},
  "current_action_instance_id": "5e578abedc44840015c8096c",
  "stack_id": [
    "5e574feaf63b0a001564e6db"
  ],
  "last_plan_action_instance_id": "5e578a95dc44840015c8096b",
  "last_plan_action_instance_status": "Active",
  "InputParameters": {
    "input_parameters": [
      {
        "name": "nsxt_transport_zone",
        "label": "Tansport Zone Name",
        "customtype": "string",
        "type": "string",
        "immutable": false,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The name of the Transport Zone to be used by the logical switch.",
        "default": "CAMDC2Zone",
        "editable": true,
        "value": "CAMDC2Zone"
      },
      {
        "name": "url",
        "label": "Rest URL",
        "customtype": "string",
        "type": "string",
        "immutable": false,
        "hidden": false,
        "required": false,
        "secured": false,
        "isDynamic": false,
        "editable": true,
        "default": "https://jsonplaceholder.typicode.com/todos/1",
        "value": "https://jsonplaceholder.Invalid"
      }
    ],
    "pre_action_input_parameters": [],
    "post_action_input_parameters": []
  }
}
```

Edit Service Instance

The **updateServiceInstance** command edits the service instance parameters.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json"
-H "Accept: application/json"
-d @cc_create.json
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/{{SERVICE_INSTANCE_ID}}/update?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

POST call

Headers:

```
"Authorization": <bearer token>

"Content-Type": "applcation/json"
```

cc_create.json

```
{
"update_details": {
    "instance_parameters": {
    "in1": "https://jsonplaceholder.typicode.com/todos/22"
  }
}
}
```

<service_instance_id> - It is the ID of the service instance for which you want to edit.

You must provide all the updated values of service instance parameters to edit this service instance.

Retry Service Instance

The **retryServiceInstance** command attempts to retry all terminate service instances that failed.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create.json
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/${SERVICE_INSTANCE_ID}/retry?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

POST call

Headers:

```
"Authorization": <bearer token>

"Content-Type": "applcation/json"
```

cc_create.json

```
{
"retry_details": {
  "is_partial": false,
  "instance_parameters": {
    "in1": "https://jsonplaceholder.typicode.com/todos/22"
  }
}
}
```

<service_instance_id> - It is the ID of the service instance for which you want to retry.

You can run retry with all the updated values for service instance parameters. If the updated service instance parameters are not provided, the values of the previous deployments are used.

If **is_partial** not provided, then by default its value is false. Support is not available for a true value of **is_partial**.

Resume Service Instance

The **resumeServiceInstance** command attempts to resume all terminated service instances that failed.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create.json
```

```
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/${SERVICE_INSTANCE_ID}/resume?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

POST call

Headers:

```
"Authorization”: <bearer token>

"Content-Type”: "application/json"
```

Sample cc_create.json:

```
{
"resume_details”: {
  "is_partial”: false,
  "instance_parameters”: {
    "in1": "https://jsonplaceholder.typicode.com/todos/22”
  }
}
}
```

<service_instance_id> - It is the ID of the service instance for which you want to resume.

You can resume deployment with all the updated values for service instance parameters. If the updated service instance parameters are not provided, the values of the previous deployments are used.

If **is_partial** not provided, then by default its value is false. Support is not available for a true value of **is_partial**.

Upgrade or downgrade a Service Instance

To upgrade or downgrade a particular service instance to higher or lower service version, the following APIs must be used in sequence:

- getServiceInstance: Returns the service instance details along with the available version change paths (upgrade/downgrade).
- versionChangeInputParameters: Returns the list of input parameters that are needed for service instance version change.
- submitversionChange: Upgrade/downgrade a version of service instance.

getServiceInstance

This API returns the service instance details along with the available version change paths (upgrade or downgrade).

```
curl --location --request GET \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/5e574fe2dc44840015c8093b?
filter[include]=UpgradePaths&filter[include]=DowngradePaths&cloudOE_spaceGuid=development&ace_orgGuid=all&ten
antId=c26b3dd4-4e74-4545-8ac9-bc30a6167b33' \
--header 'Content-Type: application/json'
```

Response

```
{
  {
  "name": "switch_resthookIns",
  "Status": "Error",
  "owner": "admin",
  "active_time": "2020-02-27T05:14:10+00:00",
  "templateName": "",
  "ordered_time": "2020-02-27T05:13:06.943Z",
  "updated_time": "2020-02-27T09:24:47.020Z",
  "creationTime": "2020-02-27T05:13:06.943Z",
  "instance_id": "d7395b20-591f-11ea-bab8-5f5432f25af5",
  "activity_deployment_details": {
    "instance": {
      "vmwarens666346e3": {
        "action": "CREATE",
        "activity_id": "vmwarens666346e3",
        "connection_name": "NSX",
        "created_at": "2020-02-27T05:13:14.914Z",
        "fail_quietly": "false",
        "flow_control_#": "1",
        "id": "5e574feaf63b0a001564e6db",
        "instance_name": "switch_resthookIns-00-inst",
        "long_running_delay": "0",
        "long_running_timeout": "0",
        "provider_engine_version": "0.11.11",
        "resource_id": "serviceblueprint_stack.vmwarens666346e3",
        "sequence_no": "1",
        "stack_data_%": "1",
        "stack_data_latest_state_id": "5e574febf63b0a001564e6dc",
        "stack_error_message": "",
        "stack_id": "5e574feaf63b0a001564e6db",
```

```
        "stack_url": "",
        "status": "SUCCESS",
        "status_error_message": "",
        "status_message": "[SUCCESS] ",
        "template_name": "VMware NSX-T Sample to create a logical switch",
        "template_outputs_%": "0",
        "template_params_%": "7",
        "template_params_nsxt_logical_switch_desc": "des",
        "template_params_nsxt_logical_switch_name": "name",
        "template_params_nsxt_logical_switch_repl_mode": "MTEP",
        "template_params_nsxt_logical_switch_state": "UP",
        "template_params_nsxt_tag": "Auto",
        "template_params_nsxt_tag_scope": "Auto-Project",
        "template_params_nsxt_transport_zone": "CAMDC2Zone",
        "template_provider": "VMware NSX-T",
        "template_tags_%": "4",
        "template_tags_request_group": "admin",
        "template_tags_request_user": "admin",
        "template_tags_service_identifier": "5e574fe2dc44840015c8093b",
        "template_tags_service_name": "switch_resthookins",
        "template_type": "TERRAFORM",
        "template_version": "2.3",
        "service_action_type": "provision",
        "service_action_name": "Provision",
        "action_instance_name": "switch_resthookIns",
        "state": "CREATED",
        "flow_control_0.case": "",
        "flow_control_0.case_is_selected": "",
        "flow_control_0.continue_failed": "false",
        "flow_control_0.continue_on_prev_failed": "false",
        "flow_control_0.continue_on_prev_inprogress": "false",
        "flow_control_0.depends_on.#": "1",
        "flow_control_0.depends_on.0": "serviceblueprint_action_start.",
        "flow_control_0.meta.%": "0",
        "flow_control_0.prev_resource_sequence_no": "0",
        "flow_control_0.prev_resource_status": "SUCCESS",
        "flow_control_0.resume": "false",
        "flow_control_0.resume_version": "0",
        "flow_control_0.version": "0"
      }
    },
    "output_parameters": {}
  },
  "logs": "serviceblueprint_action_start.start: Refreshing state... (ID:
provision)\nserviceblueprint_stack.vmwarens666346e3: Refreshing state... (ID:
5e574feaf63b0a001564e6db)\nserviceblueprint_http.resthook2ec4d4f7: Refreshing state... (ID:
resthook2ec4d4f7)\nserviceblueprint_http.resthook16dc50cc: Refreshing state... (ID:
resthook16dc50cc)\nserviceblueprint_http.resthook16dc50cc: Destroying... (ID:
resthook16dc50cc)\nserviceblueprint_http.resthook2ec4d4f7: Modifying... (ID: resthook2ec4d4f7)\n  url:
\"https://jsonplaceholder.typicode.com/todos/1\" =>
\"https://jsonplaceholder.Invalid\"\nserviceblueprint_http.resthook2ec4d4f7: Modifications complete after 0s
(ID: resthook2ec4d4f7)\nserviceblueprint_http.resthook16dc50cc: Still destroying... (ID: resthook16dc50cc,
10s elapsed)\nserviceblueprint_http.resthook16dc50cc: Destruction complete after 11s\n\nApply complete!
Resources: 0 added, 1 changed, 1 destroyed.\n\nOutputs:\n\nresthook2ec4d4f7_message = { \"resthook_88f097\":
{ \"status\": \"FAILED\" ,  \"code\": \"502\" ,  \"message\": \"Failure caused by : <nil>\"
}}\nresthook2ec4d4f7_status = FAILED\nresthook2ec4d4f7_status_error_message = <nil>\nvmwarens666346e3_message
= { \"switch_resthookIns-00-inst\": { \"status\": \"SUCCESS\" ,  \"action\": \"CREATE\"
}}\nvmwarens666346e3_status = SUCCESS\nvmwarens666346e3_status_create_error_message =
\nvmwarens666346e3_status_error_message =\n",
  "serviceId": "5e4f59ca826bc70015dfaec8",
  "serviceManifestId": "5e4f5a51826bc70015dfaece",
  "ownership": {
    "username": "admin",
    "displayName": "admin",
    "orgGuid": "all",
    "spaceGuid": "development",
    "tenantId": "c26b3dd4-4e74-4545-8ac9-bc30a6167b33"
  },
  "namespaceId": "development",
  "tenantId": "c26b3dd4-4e74-4545-8ac9-bc30a6167b33",
  "instance_plan": "Standard",
  "action": "Provision",
  "id": "5e574fe2dc44840015c8093b",
  "ServiceID": "5e4f59ca826bc70015dfaec8",
  "instance_plan_id": "5e57566fdc44840015c8094c",
  "resourceProperties": {
    "templateParameters": [],
    "specifications": [
      {
        "label": "Tansport Zone Name",
        "name": "nsxt_transport_zone",
```

```
        "value": "CAMDC2Zone",
        "secured": false
      },
      {
        "label": "Rest URL",
        "name": "url",
        "value": "https://jsonplaceholder.Invalid",
        "secured": false
      }
    ]
  },
  "catalogItemId": "5e4f59ca826bc70015dfaec8",
  "tags_input": [
    {
      "value": "admin",
      "label": "request_group",
      "name": "request_group",
      "secured": false
    },
    {
      "value": "admin",
      "label": "request_user",
      "name": "request_user",
      "secured": false
    },
    {
      "value": "switch_resthookins",
      "label": "service_name",
      "name": "service_name",
      "secured": false
    },
    {
      "value": "5e574fe2dc44840015c8093b",
      "label": "service_identifier",
      "name": "service_identifier",
      "secured": false
    }
  ],
  "ace_orgGuid": "all",
  "cloudOE_spaceGuid": "development",
  "Bearer_Token": "*******",
  "bindable": false,
  "catalogItemName": "switch_resthook",
  "serviceManifest": {
    "version": "2.2.2.2",
    "ServiceEngine_InputFile": {
      "service": {
        "version": "2.2.2.2",
        "specVersion": "v3",
        "catalog_metadata": {
          "name": "switch_resthook",
          "description": "",
          "image": "serviceicon_1.svg",
          "category": "CloudServices",
          "bullets": [],
          "providerDisplayName": "IBM",
          "longDescription": "",
          "documentationUrl": "",
          "supportUrl": "",
          "bindable": false,
          "updatable": "",
          "systemTags": true
        },
        "tags": [
          {
            "name": "request_group",
            "label": "request_group",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The current context group id of the current user that requested the provisioning
of the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
            "isSystemTag": true,
            "default": "${svc_instance.group}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
```

```
            "name": "request_user",
            "label": "request_user",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The user id of the current user that requested the provisioning of the
template.",
            "isSystemTag": true,
            "default": "${svc_instance.owner}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
            "name": "service_name",
            "label": "service_name",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Name provided for the service instance by the end user at time of request",
            "isSystemTag": true,
            "default": "${svc_instance.name}",
            "customtype": "string",
            "permission": "Read-Only"
          },
          {
            "name": "service_identifier",
            "label": "service_identifier",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
            "isSystemTag": true,
            "default": "${svc_instance.service_instance_id}",
            "customtype": "string",
            "permission": "Read-Only"
          }
        ],
        "upgrade_from": [
          {
            "version": "1.1.1.1"
          }
        ],
        "downgrade_to": [
          {
            "version": "1.1.1.1"
          }
        ],
        "actions": [
          {
            "type": "provision",
            "name": "Provision",
            "description": "Default action for deployment of service",
            "input_parameters": [
              {
                "name": "nsxt_transport_zone",
                "label": "Tansport Zone Name",
                "customtype": "string",
                "type": "string",
                "immutable": false,
                "hidden": false,
                "required": false,
                "secured": false,
                "description": "The name of the Transport Zone to be used by the logical switch.",
                "default": "CAMDC2Zone",
                "editable": true
              },
              {
                "name": "url",
                "label": "Rest URL",
                "customtype": "string",
                "type": "string",
                "immutable": false,
                "hidden": false,
                "required": false,
```

```
              "secured": false,
              "isDynamic": false,
              "editable": true,
              "default": "https://jsonplaceholder.typicode.com/todos/1"
            }
          ],
          "flow": {
            "conditions": [],
            "templates": [
              {
                "VMware NSX-T Sample to create a logical switch": {
                  "title": "VMware NSX-T Sample to create a logical switch_b70a6e",
                  "template_name": "VMware NSX-T Sample to create a logical switch",
                  "version": "2.3",
                  "id": "vmwarens666346e3",
                  "template_type": "Terraform",
                  "template_content_type": "VMware NSX-T",
                  "template_provider": "VMware NSX-T",
                  "instance_name": "",
                  "cloud_connection_name": "NSX",
                  "template_data_objects": {},
                  "template_params": {
                    "nsxt_tag_scope": "Auto-Project",
                    "nsxt_tag": "Auto",
                    "nsxt_transport_zone": "CAMDC2Zone",
                    "nsxt_logical_switch_name": "name",
                    "nsxt_logical_switch_desc": "des",
                    "nsxt_logical_switch_state": "UP",
                    "nsxt_logical_switch_repl_mode": "MTEP"
                  },
                  "outputs": [],
                  "editable": true,
                  "provider_engine_version": "0.11.11"
                }
              }
            ],
            "resthooks": [
              {
                "resthook": {
                  "title": "resthook_88f097",
                  "id": "resthook2ec4d4f7",
                  "headers": {
                    "accept": "application/json",
                    "Content-Type": "application/json"
                  },
                  "editable": true,
                  "url": "${input_parameters.url}",
                  "method": "GET"
                }
              }
            ],
            "notifications": [],
            "sequence": {
              "0": "vmwarens666346e3",
              "1": "resthook2ec4d4f7"
            },
            "error_sequence": {}
          },
          "output_parameters": []
        }
      ],
      "plans": [
        {
          "name": "Standard",
          "description": "To deploy a Standard plan",
          "actions": [],
          "plan_parameters": []
        }
      ]
    }
  },
  "service_id": "5e4f59ca826bc70015dfaec8",
  "tenantId": "c26b3dd4-4e74-4545-8ac9-bc30a6167b33",
  "namespaceId": "",
  "author": "admin",
  "specVersion": "v3",
  "status": "Draft",
  "tags": [
    {
      "name": "request_group",
      "label": "request_group",
```

```
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The current context group id of the current user that requested the provisioning of
the template. This is important because a user may be part of more than one group and this identifies the
context of which group the user made the request.",
        "isSystemTag": true,
        "default": "${svc_instance.group}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "request_user",
        "label": "request_user",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "The user id of the current user that requested the provisioning of the template.",
        "isSystemTag": true,
        "default": "${svc_instance.owner}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_name",
        "label": "service_name",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Name provided for the service instance by the end user at time of request",
        "isSystemTag": true,
        "default": "${svc_instance.name}",
        "customtype": "string",
        "permission": "Read-Only"
      },
      {
        "name": "service_identifier",
        "label": "service_identifier",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
        "isSystemTag": true,
        "default": "${svc_instance.service_instance_id}",
        "customtype": "string",
        "permission": "Read-Only"
      }
    ],
    "systemTags": true,
    "creationTime": "2020-02-21T04:19:29.736Z",
    "serviceInstanceCount": 5,
    "id": "5e4f5a51826bc70015dfaece",
    "actions": [
      {
        "type": "provision",
        "name": "Provision",
        "description": "Default action for deployment of service",
        "input_parameters": [
          {
            "name": "nsxt_transport_zone",
            "label": "Tansport Zone Name",
            "customtype": "string",
            "type": "string",
            "immutable": false,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The name of the Transport Zone to be used by the logical switch.",
            "default": "CAMDC2Zone",
            "editable": true
          },
          {
```

```
            "name": "url",
            "label": "Rest URL",
            "customtype": "string",
            "type": "string",
            "immutable": false,
            "hidden": false,
            "required": false,
            "secured": false,
            "isDynamic": false,
            "editable": true,
            "default": "https://jsonplaceholder.typicode.com/todos/1"
          }
        ],
        "flow": {
          "conditions": [],
          "templates": [
            {
              "VMware NSX-T Sample to create a logical switch": {
                "title": "VMware NSX-T Sample to create a logical switch_b70a6e",
                "template_name": "VMware NSX-T Sample to create a logical switch",
                "version": "2.3",
                "id": "vmwarens666346e3",
                "template_type": "Terraform",
                "template_content_type": "VMware NSX-T",
                "template_provider": "VMware NSX-T",
                "instance_name": "",
                "cloud_connection_name": "NSX",
                "template_data_objects": {},
                "template_params": {
                  "nsxt_tag_scope": "Auto-Project",
                  "nsxt_tag": "Auto",
                  "nsxt_transport_zone": "CAMDC2Zone",
                  "nsxt_logical_switch_name": "name",
                  "nsxt_logical_switch_desc": "des",
                  "nsxt_logical_switch_state": "UP",
                  "nsxt_logical_switch_repl_mode": "MTEP"
                },
                "outputs": [],
                "editable": true,
                "provider_engine_version": "0.11.11"
              }
            }
          ],
          "resthooks": [
            {
              "resthook": {
                "title": "resthook_88f097",
                "id": "resthook2ec4d4f7",
                "headers": {
                  "accept": "application/json",
                  "Content-Type": "application/json"
                },
                "editable": true,
                "url": "${input_parameters.url}",
                "method": "GET"
              }
            }
          ],
          "notifications": [],
          "sequence": {
            "0": "vmwarens666346e3",
            "1": "resthook2ec4d4f7"
          },
          "error_sequence": {}
        },
        "output_parameters": []
      }
    ],
    "plans": [
      {
        "name": "Standard",
        "description": "To deploy a Standard plan",
        "actions": [],
        "plan_parameters": []
      }
    ],
    "tagId": "5e4f59ca826bc70015dfaec7"
  },
  "action_status": {},
  "output_parameters": [],
  "bind_array": {},
```

```
  "current_action_instance_id": "5e578abedc44840015c8096c",
  "stack_id": [
    "5e574feaf63b0a001564e6db"
  ],
  "last_plan_action_instance_id": "5e578a95dc44840015c8096b",
  "last_plan_action_instance_status": "Active",
  "actionInstanceDetails": [
    {
      "actionInstanceId": "5e574fe3dc44840015c8093c",
      "Status": "Active",
      "processed_time": "2020-02-27T05:13:07.034Z",
      "active_time": "2020-02-27T05:14:10+00:00",
      "ordered_time": 1582780387033,
      "created_time": "2020-02-27T05:14:10+00:00",
      "updated_time": "2020-02-27T05:14:10.982Z",
      "instance_action": "Provision",
      "input_parameters": {}
    },
    {
      "actionInstanceId": "5e575621dc44840015c80943",
      "Status": "Error",
      "processed_time": "2020-02-27T05:39:45.176Z",
      "ordered_time": 1582781985140,
      "updated_time": "2020-02-27T05:39:45.917Z",
      "instance_action": "plan",
      "versionChangeDetails": {
        "versionChange": true,
        "versionChangeId": "5e50fc1405789a0015a482f5",
        "instance_parameters": {
          "nsxt_transport_zone": "CAMDC2Zone",
          "url": "https://jsonplaceholder.typicode.com/todos/1"
        },
        "pre_action_parameters": {},
        "post_action_parameters": {},
        "fromVersionManifest": {},
        "isRegenerationRequired": true,
        "toVersion": "2.2.2.2"
      },
      "input_parameters": {
        "nsxt_transport_zone": "CAMDC2Zone",
        "url": "https://jsonplaceholder.typicode.com/todos/1"
      }
    },
    {
      "actionInstanceId": "5e57564fdc44840015c80948",
      "Status": "Error",
      "processed_time": "2020-02-27T05:40:31.198Z",
      "ordered_time": 1582782031165,
      "updated_time": "2020-02-27T05:40:31.649Z",
      "instance_action": "plan",
      "versionChangeDetails": {
        "versionChange": true,
        "versionChangeId": "5e575610dc44840015c8093f",
        "instance_parameters": {
          "nsxt_transport_zone": "CAMDC2Zone",
          "url": "https://jsonplaceholder.typicode.com/todos/1"
        },
        "pre_action_parameters": {},
        "post_action_parameters": {},
        "fromVersionManifest": {},
        "isRegenerationRequired": true,
        "toVersion": "3.3.3.3"
      },
      "input_parameters": {
        "nsxt_transport_zone": "CAMDC2Zone",
        "url": "https://jsonplaceholder.typicode.com/todos/1"
      }
    },
    {
      "actionInstanceId": "5e5756b8dc44840015c80954",
      "Status": "Active",
      "processed_time": "2020-02-27T05:42:16.162Z",
      "active_time": "2020-02-27T05:42:47+00:00",
      "ordered_time": 1582782136137,
      "created_time": "2020-02-27T05:42:47+00:00",
      "updated_time": "2020-02-27T05:42:47.053Z",
      "instance_action": "plan",
      "versionChangeDetails": {
        "versionChange": true,
        "versionChangeId": "5e57566fdc44840015c8094d",
        "instance_parameters": {
```

```
        "nsxt_transport_zone": "CAMDC2Zone",
        "url": "https://jsonplaceholder.typicode.com/todos/1"
      },
      "pre_action_parameters": {},
      "post_action_parameters": {},
      "fromVersionManifest": {},
      "isRegenerationRequired": true,
      "toVersion": "2.2.2.2"
    },
    "input_parameters": {
      "nsxt_transport_zone": "CAMDC2Zone",
      "url": "https://jsonplaceholder.typicode.com/todos/1"
    }
  },
  {
    "actionInstanceId": "5e5756eadc44840015c80955",
    "Status": "Active",
    "processed_time": "2020-02-27T05:43:06.719Z",
    "active_time": "2020-02-27T05:43:38+00:00",
    "ordered_time": 1582782186716,
    "created_time": "2020-02-27T05:43:38+00:00",
    "updated_time": "2020-02-27T05:43:38.994Z",
    "instance_action": "update",
    "versionChangeDetails": {
      "serviceInstanceId": "5e574fe2dc44840015c8093b",
      "versionChangeId": "5e57566fdc44840015c8094d",
      "versionChangeType": "UPGRADE",
      "isRegenerationRequired": true,
      "fromVersionId": "5e4f59ca826bc70015dfaec9",
      "fromVersion": "1.1.1.1",
      "fromVersionManifest": {},
      "toVersionId": "5e4f5a51826bc70015dfaece",
      "toVersion": "2.2.2.2",
      "instance_parameters": {
        "nsxt_transport_zone": "CAMDC2Zone",
        "url": "https://jsonplaceholder.typicode.com/todos/1"
      },
      "pre_action_ParmList": [],
      "pre_action_parameters": {},
      "post_action_ParmList": [],
      "post_action_parameters": {}
    },
    "input_parameters": {
      "nsxt_transport_zone": "CAMDC2Zone",
      "url": "https://jsonplaceholder.typicode.com/todos/1"
    }
  },
  {
    "actionInstanceId": "5e57891bdc44840015c80969",
    "Status": "Active",
    "processed_time": "2020-02-27T09:17:15.511Z",
    "active_time": "2020-02-27T09:17:47+00:00",
    "ordered_time": 1582795035480,
    "created_time": "2020-02-27T09:17:47+00:00",
    "updated_time": "2020-02-27T09:17:47.695Z",
    "instance_action": "plan",
    "versionChangeDetails": {
      "versionChange": true,
      "versionChangeId": "5e575d04dc44840015c80963",
      "instance_parameters": {
        "url": "https://jsonplaceholder.typicode.com/todos/1",
        "url2": "https://jsonplaceholder.typicode.com/todos/2"
      },
      "pre_action_parameters": {},
      "post_action_parameters": {},
      "fromVersionManifest": {},
      "isRegenerationRequired": true,
      "toVersion": "3.3.3.3"
    },
    "input_parameters": {
      "url": "https://jsonplaceholder.typicode.com/todos/1",
      "url2": "https://jsonplaceholder.typicode.com/todos/2"
    }
  },
  {
    "actionInstanceId": "5e578990dc44840015c8096a",
    "Status": "Active",
    "processed_time": "2020-02-27T09:19:12.978Z",
    "active_time": "2020-02-27T09:19:46+00:00",
    "ordered_time": 1582795152977,
    "created_time": "2020-02-27T09:19:46+00:00",
```

```
      "updated_time": "2020-02-27T09:19:46.727Z",
      "instance_action": "update",
      "versionChangeDetails": {
        "serviceInstanceId": "5e574fe2dc44840015c8093b",
        "versionChangeId": "5e575d04dc44840015c80963",
        "versionChangeType": "UPGRADE",
        "isRegenerationRequired": true,
        "fromVersionId": "5e4f5a51826bc70015dfaece",
        "fromVersion": "2.2.2.2",
        "fromVersionManifest": {},
        "toVersionId": "5e575610dc44840015c8093d",
        "toVersion": "3.3.3.3",
        "instance_parameters": {
          "url": "https://jsonplaceholder.typicode.com/todos/1",
          "url2": "https://jsonplaceholder.typicode.com/todos/2"
        },
        "pre_action_ParmList": [],
        "pre_action_parameters": {},
        "post_action_ParmList": [],
        "post_action_parameters": {}
      },
      "input_parameters": {
        "url": "https://jsonplaceholder.typicode.com/todos/1",
        "url2": "https://jsonplaceholder.typicode.com/todos/2"
      }
    },
    {
      "actionInstanceId": "5e578a95dc44840015c8096b",
      "Status": "Active",
      "processed_time": "2020-02-27T09:23:33.091Z",
      "active_time": "2020-02-27T09:24:05+00:00",
      "ordered_time": 1582795413060,
      "created_time": "2020-02-27T09:24:05+00:00",
      "updated_time": "2020-02-27T09:24:05.271Z",
      "instance_action": "plan",
      "versionChangeDetails": {
        "versionChange": true,
        "versionChangeId": "5e575d04dc44840015c80964",
        "instance_parameters": {
          "nsxt_transport_zone": "CAMDC2Zone",
          "url": "https://jsonplaceholder.Invalid"
        },
        "pre_action_parameters": {},
        "post_action_parameters": {},
        "fromVersionManifest": {},
        "isRegenerationRequired": true,
        "toVersion": "2.2.2.2"
      },
      "input_parameters": {
        "nsxt_transport_zone": "CAMDC2Zone",
        "url": "https://jsonplaceholder.Invalid"
      }
    },
    {
      "actionInstanceId": "5e578abedc44840015c8096c",
      "Status": "Error",
      "processed_time": "2020-02-27T09:24:14.187Z",
      "ordered_time": 1582795454186,
      "updated_time": "2020-02-27T09:24:47.020Z",
      "instance_action": "update",
      "versionChangeDetails": {
        "serviceInstanceId": "5e574fe2dc44840015c8093b",
        "versionChangeId": "5e575d04dc44840015c80964",
        "versionChangeType": "DOWNGRADE",
        "isRegenerationRequired": true,
        "fromVersionId": "5e575610dc44840015c8093d",
        "fromVersion": "3.3.3.3",
        "fromVersionManifest": {},
        "toVersionId": "5e4f5a51826bc70015dfaece",
        "toVersion": "2.2.2.2",
        "instance_parameters": {
          "nsxt_transport_zone": "CAMDC2Zone",
          "url": "https://jsonplaceholder.Invalid"
        },
        "pre_action_ParmList": [],
        "pre_action_parameters": {},
        "post_action_ParmList": [],
        "post_action_parameters": {}
      },
      "input_parameters": {
        "nsxt_transport_zone": "CAMDC2Zone",
```

```
        "url": "https://jsonplaceholder.Invalid"
      }
    }
  ],
  "processed_time": "2020-02-27T09:24:14.187Z",
  "output": {},
  "UpgradePaths": [
    {
      "currentVersionId": "5e575610dc44840015c8093d",
      "fromVersionId": "5e4f5a51826bc70015dfaece",
      "toVersionId": "5e575610dc44840015c8093d",
      "type": "UPGRADE",
      "tenantId": "c26b3dd4-4e74-4545-8ac9-bc30a6167b33",
      "id": "5e575d04dc44840015c80963"
    }
  ],
  "DowngradePaths": [
    {
      "currentVersionId": "5e4f5a51826bc70015dfaece",
      "fromVersionId": "5e4f5a51826bc70015dfaece",
      "toVersionId": "5e4f59ca826bc70015dfaec9",
      "type": "DOWNGRADE",
      "tenantId": "c26b3dd4-4e74-4545-8ac9-bc30a6167b33",
      "id": "5e57566fdc44840015c8094e"
    }
  ]
}
```

Note: **id** present inside the UpgradePaths or DowngradePaths should be used as **versionChangeId** in payloads of the **versionChangeInputParameters** and **submitVersionChange** APIs.

versionChangeInputParameters

This API returns the list of input parameters that are needed for service instance version change.

```
curl --location --request POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/5e574fe2dc44840015c8093b/versionChangeInputPara
meters?cloudOE_spaceGuid=development&ace_orgGuid=all&tenantId=c26b3dd4-4e74-4545-8ac9-bc30a6167b33' \
--header 'Content-Type: application/json' \
--data-raw '{
  "versionChangeId": "5e575d04dc44840015c80963"
}
```

versionChangeId is the ID that is present inside the **UpgradePaths** or **DowngradePaths**, which is returned with the **getServiceInstance** API.

Request Body

```
{
  "versionChangeId": "5e575c44dc44840015c80959"
}
```

Response

```
{
  "VersionChangeInputParameters": {
    "versionChangeId": "5e575d04dc44840015c80963",
    "toVersion": "3.3.3.3",
    "type": "UPGRADE",
    "input_parameters": [
      {
        "name": "url",
        "label": "Rest URL",
        "customtype": "string",
        "type": "string",
        "immutable": false,
        "hidden": false,
        "required": false,
        "secured": false,
        "isDynamic": false,
        "editable": true,
        "default": "https://jsonplaceholder.Invalid"
      },
      {
        "name": "url2",
        "label": "URL 2",
        "customtype": "string",
        "type": "string",
        "immutable": false,
        "hidden": false,
```

```
        "required": false,
        "secured": false,
        "isDynamic": false,
        "editable": true,
        "default": "https://jsonplaceholder.typicode.com/todos/2"
      }
    ],
    "pre_action_input_parameters": [],
    "post_action_input_parameters": [],
    "new_parameters": [
      "url2"
    ],
    "used_old_parameters": [
      "url"
    ],
    "ignored_old_parameters": [
      "nsxt_transport_zone"
    ]
  }
}
```

Note:

- `input_parameters` - A list of input parameters needed for service instance version change.
- `new_parameters` - A list of new input parameters present in the new version.
- `used_old_parameters`- A list of input parameters present from the old version which are used in new version.
- `ignored_old_parameters` - A list of input parameters present from the old version which are not used in new version.
- `pre_action_input_parameters` and `post_action_input_parameters` are not supported in this release.

submitVersionChange

This API upgrades/downgrades a version of the service instance.

```
curl --location --request POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/5e574fe2dc44840015c8093b/submitVersionChange?
cloudOE_spaceGuid=development&ace_orgGuid=all&tenantId=c26b3dd4-4e74-4545-8ac9-bc30a6167b33' \
--header 'Content-Type: application/json' \
--data-raw '{
    "versionChangeId": "5e575d04dc44840015c80963",
    "instance_parameters": {
        "url": "https://jsonplaceholder.typicode.com/todos/1",
        "url2": "https://jsonplaceholder.typicode.com/todos/2"
    },
    "pre_action_parameters": {},
    "post_action_parameters": {},
    "plan": ""
}
```

Request Body

```
{
    "versionChangeId": "5e575d04dc44840015c80963",
    "instance_parameters": {
        "url": "https://jsonplaceholder.typicode.com/todos/1",
        "url2": "https://jsonplaceholder.typicode.com/todos/2"
    },
    "pre_action_parameters": {},
    "post_action_parameters": {},
    "plan": ""
}
```

Bind a service instance

The API call creates a bind instance.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create.json
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/${SERVICE_INSTANCE_ID}/createBind?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

POST call

Headers:

```
"Authorization":  "bearer <token>"

"Content-Type": "application/json"
```

cc_create.json

```
{
 "bindDetails": {
   "instance_parameters": {
     "resturl2": "https://jsonplaceholder.typicode.com/todos/33""
   }
 }
```

- **<service_instance_id>** - It is the ID of the service instance for which you want to create a bind instance.
- **<instance_parameters>** - Optionally, you can run bind with updated values for service instance parameters. If not provided, then earlier values are used.

Unbind Instance

The API call unbinds or deletes a bind instance.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create.json
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/${SERVICE_INSTANCE_ID}/terminateBind?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

BODY:

cc_create.json

```
"name":"<name_of_bind_instance>"}
```

<name_of_bind_instance> - It is the name of the bind instance that you want to unbind.

Get custom actions

List the custom actions of a service instance.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json"
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/<serviceInstanceID>/getCustomActions?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

This GET call fetches the list of custom actions of service instance.

```
Headers:

"Authorization": "bearer <token>"
"Content-Type": "application/json"
```

Response

```
{
  "customActions": [
    {
      "type": "custom",
      "name": "custom1",
      "description": "customaction",
      "pre_or_post": false,
      "input_parameters": [
        {
          "name": "resthook_url",
          "label": "resthook_url",
          "customtype": "string",
          "type": "string",
          "immutable": false,
          "hidden": false,
          "required": false,
          "secured": false,
          "isDynamic": false
        }
      ],
      "flow": {
        "conditions": [],
        "templates": [],
        "resthooks": [
          {
            "resthook": {
              "title": "resthook_f428a5",
```

```
              "id": "resthook56a3f1fc",
              "headers": {
                "accept": "application/json",
                "Content-Type": "application/json"
              },
              "editable": true,
              "url": "${input_parameters.resthook_url}",
              "method": "GET"
            }
          }
        ],
        "notifications": [],
        "sequence": {
          "0": "resthook56a3f1fc"
        },
        "error_sequence": {}
      },
      "output_parameters": []
    }
  ]
}
```

Invoke custom action

To execute a custom action on a service instance.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @custom_action.json
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/<serviceInstanceID>/invokeCustomAction?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

POST call

```
Headers:

"Authorization": "bearer <token>"
"Content-Type": "application/json"
```

custon_action.json

```
{
  "customDetails": {
    "serviceInstanceId": "<serviceInstanceId>"
    "name":"actionInstanceName",
    "instance_parameters": {
      "resthook_url":"https://jsonplaceholder.typicode.com/todos/33"
    },
    "currentCustomAction":"<actionName>"
  }
}
```

instance_parameters - Optionally, you can run custom action with updated values for service instance parameters. The default values are used, if parameters are not provided.

Terminate Service Instance

```
curl -k -X POST \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/${ServiceInstanceId}/terminate?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
-H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
-H 'Content-Type: application/json' \
-H 'cache-control: no-cache' \
-d @service__instance_terminate.json
```

Content of service__instance_terminate.json:

```
{}
```

Response

```
{"name":"upgarde_downgrade_service_instance56857d10-0220-11eb-859b-
adea70bdca6d","Status":"Terminating","id":"5f72d9ac2401e29758629ce9"}
```

Cleanup Service Instance

It deletes all service instances based on the value of **force** flag. If the **force = false**, then the instances in active state are not deleted. The instances in error state are deleted irrespective of the **force** flag value.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create.json
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/{{SERVICE_INSTANCE_ID}}/cleanup?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"

POST call

Headers:

"Authorization": <bearer token>
"Content-Type": "applcation/json"

Body: {
 "force": <true/false>
}
```

Cleanup Service Instance in inprogress state

If a service instance is stuck in IN_PROGRESS state for a long time without any backend activity, then use this API to delete the service instance entry from the database.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create.json
"https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceInstances/{{SERVICE_INSTANCE_ID}}/cleanup?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Sample Request BODY:

```
{
  "serviceInstanceId": "5cf0c20cbdf17d001786d1a9",
  "force": true
}
```

The **`force = true`** deletes the service instance entry from the Managed services database forcefully. However, the activities that are running does not get cleaned up and they throw the following error exception:

```
{
  "error": {
    "statusCode": "419",
    "name": "Error",
    "message": "ResourceLockError: Failed to obtain lock for resource stack_e19dd5d0-85f5-11e9-8609-
cf2b57a8a189.; caused by Error: unable to obtain lock"
  }
}
```

The **`ResourceLockError`** is related to Mongo DB lock. Whenever any deployment starts, Mongo DB lock associated with the stack_id is placed and it remains locked for 1 hour. If service instance is stuck and the cleanup API gives ResourceLockError, try the same API after 1 hour, that is, after the lock expires. It is a precautionary step to avoid cleaning up of service instances that are in running state at the backend, which might cause inconsistency. Also, there can be side effects if an instance having dangled resources gets cleaned up.

Standalone action APIs

Create Standalone Action

Creates a stand-alone action independent of any service.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" -H "Accept: application/json" \
-d @cc_create.json "https://${TSA_HOSTNAME}/cam/composer/api/v1/Actions/createAction?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid={NAMESPACE}"
```

The **`cloudOE_spaceGuid`** contains the value of the namespace. Note that the **`NAMESPACE`** must be the desired namespace to deploy to.

POST method

Example BODY:

cc_create.json

```
{
    "type": "provision",
```

```
    "name": "newAct44f",
    "description": "Default action for deployment of service",
    "input_parameters": [],
    "flow": {
        "conditions": [],
        "templates": [
            {
                "Template With Output Parameters": {
                    "title": "Template With Output Parameters_343311",
                    "template_name": "Template With Output Parameters ce0c21",
                    "version": "master",
                    "id": "template46b48313",
                    "template_type": "Terraform",
                    "template_content_type": "VMware vSphere",
                    "template_provider": "VMware vSphere",
                    "instance_name": "",
                    "cloud_connection_name": "Global-VV1",
                    "template_data_objects": {},
                    "template_params": {
                        "sleep": "10000",
                        "cam-ip": "9.5.29.229",
                        "data-server": "9.5.28.159",
                        "data-as-query-string": "id=1&name=test",
                        "region": "us-east-1"
                    },
                    "outputs": [
                        "resource-id",
                        "selected-region",
                        "data"
                    ]
                }
            }
        ],
        "resthooks": [],
        "notifications": [],
        "sequence": {
            "0": "template46b48313"
        }
    },
    "output_parameters": []
}
```

After your create an action, an action ID is generated automatically:

Example response:

```
{
   "ServiceEngine_InputFile": {
      "service": {
         "specVersion": "v3",
         "catalog_metadata": {
            "name": "dsfdfs",
            "description": "",
            "image": "serviceicon_1.svg",
            "category": "CloudServices",
            "bullets": [],
            "providerDisplayName": "IBM",
            "longDescription": "",
            "documentationUrl": "",
            "supportUrl": "",
            "bindable": false,
            "updatable": "",
            "systemTags": true,
            "author": "admin"
         },
         "tags": [
            {
               "name": "Request_Group",
               "label": "Request_Group",
               "type": "string",
               "immutable": true,
               "hidden": false,
               "required": false,
               "secured": false,
               "description": "The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and this
identifies the context of which group the user made the request.",
               "isSystemTag": true,
               "default": "${svc_instance.group}",
               "customtype": "string",
               "permission": "Read-Only"
```

```
            },
            {
               "name": "Request_User",
               "label": "Request_User",
               "type": "string",
               "immutable": true,
               "hidden": false,
               "required": false,
               "secured": false,
               "description": "The user id of the current user that requested the provisioning of the
template.",
               "isSystemTag": true,
               "default": "${svc_instance.owner}",
               "customtype": "string",
               "permission": "Read-Only"
            },
            {
               "name": "Service_Name",
               "label": "Service_Name",
               "type": "string",
               "immutable": true,
               "hidden": false,
               "required": false,
               "secured": false,
               "description": "Name provided for the service instance by the end user at time of request",
               "isSystemTag": true,
               "default": "${svc_instance.name}",
               "customtype": "string",
               "permission": "Read-Only"
            },
            {
               "name": "Service_Identifier",
               "label": "Service_Identifier",
               "type": "string",
               "immutable": true,
               "hidden": false,
               "required": false,
               "secured": false,
               "description": "Unique identifer generated by ICAM which is mapped to the service instance of
the template provisioned.",
               "isSystemTag": true,
               "default": "${svc_instance.service_instance_id}",
               "customtype": "string",
               "permission": "Read-Only"
            }
         ],
         "actions": [
            {
               "type": "provision",
               "name": "Provision",
               "description": "Default action for deployment of service",
               "input_parameters": [],
               "flow": {
                  "conditions": [],
                  "templates": [
                     {
                        "Amazon EC2 Virtual Server with SSH key": {
                           "title": "Amazon EC2 Virtual Server with SSH key_25e3a0",
                           "template_name": "Amazon EC2 Virtual Server with SSH key",
                           "version": "2.3",
                           "id": "amazoneceeb670ee",
                           "template_type": "Terraform",
                           "template_content_type": "Amazon EC2",
                           "template_provider": "Amazon EC2",
                           "instance_name": "",
                           "cloud_connection_name": "",
                           "template_data_objects": {},
                           "template_params": {
                              "": "us-east-1",
                              "aws_image": "ubuntu/images/hvm-ssd/ubuntu-trusty-14.04-amd64-server-*",
                              "aws_image_size": "t2.small",
                              "aws_ami_owner_id": "099720109477",
                              "vpc_name_tag": "",
                              "subnet_name": "",
                              "public_ssh_key_name": "",
                              "public_ssh_key": ""
                           },
                           "outputs": [],
                           "warning": true,
                           "current_action_key": 0
                        }
```

```
                    }
                ],
                "resthooks": [],
                "notifications": [],
                "sequence": {
                   "0": "amazoneceeb670ee"
                },
                "error_sequence": {}
             },
             "output_parameters": []
          }
       ],
       "plans": [
          {
             "name": "Standard",
             "description": "To deploy a Standard plan",
             "actions": [],
             "plan_parameters": []
          }
       ],
       "tagId": "5c36d79c3abef7001aead045"
    }
 },
 "service_id": "5cf5fd6e30483a00144c919f",
 "namespaceId": "",
 "id": "5cf5fd6e30483a00144c919e",
 "warning": [
    "Template doesnt exists or no access: response.statusCode: 400",
    " Template amazoneceeb670ee does not have cloud connection name.",
    "Templates- 'template_params' of type json object should have non-empty keys , template.id :
amazoneceeb670ee"
 ]
}
```

For complete format of the action, see action section in the Editing a service file section.

Invoke Standalone Action

Use the action id generated in the **`Create Action`** API call to invoke an action.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create.json "https://${TSA_HOSTNAME}/cam/composer/api/v1/ActionInstances/invokeAction?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid={NAMESPACE}"
```

The **`cloudOE_spaceGuid`** contains the value of the namespace. Note that the **`NAMESPACE`** must be the desired namespace to deploy to.

POST method

Example BODY:

```
"action_data": {
        "action_id": "5caf3b64b52b5a00182116af",
        "name": "inst2 "
    }
```

Example response:

```
{
    "actionInstance": {
        "id": "5cd01a095253de001918ce2c"
    }
}
```

It returns Action instance ID.

ICON APIs

Upload service icons

This API uploads new icons. Select the file to upload.

POST method

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}"  \
```

```
-F "icon=@PATH_TO_FILE" "https://${TSA_HOSTNAME}/cam/composer/api/v1/icons/upload?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

It returns the name and id of the image.

BODY example:

```
{
   "res": {
       "name": "serviceicon_1.svg",
       "id": "5czdb1bc1b47001cbc556"
     }
}
```

Note:

- The size of the icon file must be less than one MB.
- The name of the icon must be unique.
- The supported content types are as follows:

  CONTENT_TYPE = ['image/jpeg', 'image/gif', 'image/png','application/xml'];

- Upload images with the key name as ‘icon’.

Delete service icons

This API deletes service icons.

```
curl -k -X DELETE \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" "https://${TSA_HOSTNAME}/cam/composer/api/v1/icons/${ICON_ID}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Retrieve all service icons

This API returns metadata of all available icons.

GET method.

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" "https://${TSA_HOSTNAME}/cam/composer/api/v1/icons?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

BODY example:

```
[
   {
     "ContentTye": "application/xml",
     "name": "serviceicon_1.svg",
     "id": "5czdb1bc1b47001cbc556"
   },
   {
      "ContentTye": "application/xml",
      "name": "serviceicon_2.svg",
      "id": "5czdb1bc1b47001cbc557"
    },
    {
      "ContentTye": "application/xml",
      "name": "serviceicon_3.svg",
      "id": "SCZ5czdb1bc1b47001cbc559"
     }
]
```

You can also set the filter to retrieve the metadata of a service icon based on its name, ID, or type.

```
GET /cam/composer/api/v1/icons/?tenantId=62c6ce6e-1f62-4fe1-b176-259037a7bff2&ace_orgGuid=all&filter[where]
[name]=serviceicon_1.svg
```

BODY example:

```
{
   {
       "ContentTye": "application/xml",
       "name": "serviceicon_1.svg",
       "id": "5czdb1bc1b47001cbc556"
     }
}
```

Download service icon

This API call downloads the specified icon.

GET method

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}"
"https://${TSA_HOSTNAME}/cam/composer/api/v1/icons/name/${ICON_NAME}?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Custom action APIs

This section includes all the custom action APIs with examples.

- Create a custom action category
- Create a custom action
- Lock a custom action
- Unlock a custom action
- Get the list of custom actions
- Assign a custom action to service version (service manifest)
    - Additional information for the custom action to a service version (service manifest) API

Custom action APIs

Create a custom action category

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/customActionCategories?
ace_orgGuid=${MCM_TEAM}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d {"Name": "demo-category"}
```

Response

```
{
    "Name": "demo-category",
    "id": "621cc8e6218004001ac4bc00",
    "tenantId": "cf5c3f7d-a70d-435c-9554-55d8e3441380"
}
```

Create a custom action

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/CustomAction/createCustomAction?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @custom_action_create.json
```

custom_action_create.json`

```
{
  "action": {
    "type": "custom",
    "name": "ind-action",
    "category": "demo-category",
    "description": " Independent custom action",
    "input_parameters": [
      {
        "name": "inputurl",
        "label": "inputpurl",
        "customtype": "string",
        "type": "string",
        "immutable": false,
        "hidden": false,
        "required": false,
        "secured": false,
        "isDynamic": false,
        "default": "https://jsonplaceholder.typicode.com/todos/1"
      }
    ],
    "flow": {
```

```
      "conditions": [],
      "templates": [],
      "resthooks": [
        {
          "resthook": {
            "title": "resthook_354d4a",
            "id": "resthook98f44a65",
            "headers": {
              "accept": "application/json",
              "Content-Type": "application/json"
            },
            "editable": true,
            "url": "${input_parameters.inputurl}",
            "method": "GET"
          }
        }
      ],
      "notifications": [],
      "sequence": {
        "0": "resthook98f44a65"
      },
      "error_sequence": {}
    },
    "output_parameters": [
      {
        "name": "resthookOutput",
        "label": "resthookoutput",
        "customtype": "string",
        "type": "string",
        "immutable": true,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "",
        "default": "${resthooks.resthook98f44a65.output.body}"
      }
    ]
  }
}
```

Response

```
{
    "data": {
        "name": "ind-action",
        "ServiceEngine_InputFile": {
            "action": {
                "type": "custom",
                "name": "ind-action",
                "category": "demo-category",
                "description": " Independent custom action",
                "input_parameters": [
                    {
                        "name": "inputurl",
                        "label": "inputpurl",
                        "customtype": "string",
                        "type": "string",
                        "immutable": false,
                        "hidden": false,
                        "required": false,
                        "secured": false,
                        "isDynamic": false,
                        "default": "https://jsonplaceholder.typicode.com/todos/1"
                    }
                ],
                "flow": {
                    "conditions": [],
                    "templates": [],
                    "resthooks": [
                        {
                            "resthook": {
                                "title": "resthook_354d4a",
                                "id": "resthook98f44a65",
                                "headers": {
                                    "accept": "application/json",
                                    "Content-Type": "application/json"
                                },
                                "editable": true,
                                "url": "${input_parameters.inputurl}",
                                "method": "GET"
                            }
```

```
                        }
                    ],
                    "notifications": [],
                    "sequence": {
                        "0": "resthook98f44a65"
                    },
                    "error_sequence": {}
                },
                "output_parameters": [
                    {
                        "name": "resthookOutput",
                        "label": "resthookoutput",
                        "customtype": "string",
                        "type": "string",
                        "immutable": true,
                        "hidden": false,
                        "required": false,
                        "secured": false,
                        "description": "",
                        "default": "${resthooks.resthook98f44a65.output.body}"
                    }
                ]
            }
        },
        "categoryId": "621cc8e6218004001ac4bc00",
        "namespaceId": "cp4waiops",
        "tenantId": "cf5c3f7d-a70d-435c-9554-55d8e3441380",
        "creationTime": "2022-02-28T13:13:30.918Z",
        "lastUpdated": "2022-02-28T13:13:30.672Z",
        "status": "Unlocked",
        "id": "621cca7a218004001ac4bc03",
        "warning": []
    }
}
```

Lock a custom action

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/CustomAction/<customid>/lock?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d {}
```

Response:

```
{
    "data": {
        "name": "ind-action",
        "ServiceEngine_InputFile": {
            "action": {
                "type": "custom",
                "name": "ind-action",
                "category": "demo-category",
                "description": " Independent custom action",
                "input_parameters": [
                    {
                        "name": "inputurl",
                        "label": "inputpurl",
                        "customtype": "string",
                        "type": "string",
                        "immutable": false,
                        "hidden": false,
                        "required": false,
                        "secured": false,
                        "isDynamic": false,
                        "default": "https://jsonplaceholder.typicode.com/todos/1"
                    }
                ],
                "flow": {
                    "conditions": [],
                    "templates": [],
                    "resthooks": [
                        {
                            "resthook": {
                                "title": "resthook_354d4a",
                                "id": "resthook98f44a65",
                                "headers": {
                                    "accept": "application/json",
                                    "Content-Type": "application/json"
```

```
                                },
                                "editable": true,
                                "url": "${input_parameters.inputurl}",
                                "method": "GET"
                            }
                        }
                    ],
                    "notifications": [],
                    "sequence": {
                        "0": "resthook98f44a65"
                    },
                    "error_sequence": {}
                },
                "output_parameters": [
                    {
                        "name": "resthookOutput",
                        "label": "resthookoutput",
                        "customtype": "string",
                        "type": "string",
                        "immutable": true,
                        "hidden": false,
                        "required": false,
                        "secured": false,
                        "description": "",
                        "default": "${resthooks.resthook98f44a65.output.body}"
                    }
                ]
            }
        },
        "categoryId": "621cc8e6218004001ac4bc00",
        "namespaceId": "cp4waiops",
        "tenantId": "cf5c3f7d-a70d-435c-9554-55d8e3441380",
        "creationTime": "2022-02-28T13:13:30.918Z",
        "lastUpdated": "2022-02-28T13:20:01.513Z",
        "status": "Locked",
        "id": "621cca7a218004001ac4bc03"
    }
}
```

Get the list of custom actions

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" "https://${TSA_HOSTNAME}/cam/composer/api/v1/CustomAction?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Response

```
[
    {
        "name": "ind-action",
        "ServiceEngine_InputFile": {
            "action": {
                "type": "custom",
                "name": "ind-action",
                "category": "demo-category",
                "description": " Independent custom action",
                "input_parameters": [
                    {
                        "name": "inputurl",
                        "label": "inputpurl",
                        "customtype": "string",
                        "type": "string",
                        "immutable": false,
                        "hidden": false,
                        "required": false,
                        "secured": false,
                        "isDynamic": false,
                        "default": "https://jsonplaceholder.typicode.com/todos/1"
                    }
                ],
                "flow": {
                    "conditions": [],
                    "templates": [],
                    "resthooks": [
                        {
                            "resthook": {
                                "title": "resthook_354d4a",
                                "id": "resthook98f44a65",
                                "headers": {
                                    "accept": "application/json",
                                    "Content-Type": "application/json"
```

```
                                },
                                "editable": true,
                                "url": "${input_parameters.inputurl}",
                                "method": "GET"
                            }
                        }
                    ],
                    "notifications": [],
                    "sequence": {
                        "0": "resthook98f44a65"
                    },
                    "error_sequence": {}
                },
                "output_parameters": [
                    {
                        "name": "resthookOutput",
                        "label": "resthookoutput",
                        "customtype": "string",
                        "type": "string",
                        "immutable": true,
                        "hidden": false,
                        "required": false,
                        "secured": false,
                        "description": "",
                        "default": "${resthooks.resthook98f44a65.output.body}"
                    }
                ]
            }
        },
        "categoryId": "621cc8e6218004001ac4bc00",
        "namespaceId": "cp4waiops",
        "tenantId": "cf5c3f7d-a70d-435c-9554-55d8e3441380",
        "creationTime": "2022-02-28T13:13:30.918Z",
        "lastUpdated": "2022-02-28T13:22:09.539Z",
        "status": "Unlocked",
        "id": "621cca7a218004001ac4bc03",
        "serviceVersionCount": 0,
        "serviceVersionNames": []
    }
]
```

Assign a custom action to service version (service manifest)

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/ServiceManifest/<servicemanifestid>/assignCustomActions?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @assign_custom_actions.json
```

assign_custom_actions.json:

```
{
  "custom_action_details": [
    {
      "customActionName": "ind-action",
      "customActionId": "<custom action id>",
      "customInputParams": {
        "inputurl": "${svc_instance.output_parameters.todosUrl3}"
      }
    }
  ]
}
```

Response:

```
{
    "version": "1.0.0.0",
    "ServiceEngine_InputFile": {
        "service": {
            "version": "1.0.0.0",
            "specVersion": "v3",
            "catalog_metadata": {
                "name": "assignAService",
                "description": "",
                "image": "serviceicon_1.svg",
                "category": "dinesh",
                "bullets": [],
                "providerDisplayName": "IBM",
                "longDescription": "",
```

```
                "documentationUrl": "",
                "supportUrl": "",
                "bindable": false,
                "updatable": "",
                "systemTags": true
            },
            "tags": [
                {
                    "name": "environment",
                    "label": "Environment",
                    "type": "string",
                    "immutable": false,
                    "hidden": false,
                    "required": true,
                    "secured": false,
                    "description": "Unique identifer added to the provisioned VMs.",
                    "isSystemTag": true,
                    "customtype": "string",
                    "permission": "Read-Write",
                    "options": [
                        {
                            "label": "Development",
                            "value": "dev",
                            "default": true
                        },
                        {
                            "label": "Quality Assurance",
                            "value": "qa",
                            "default": false
                        },
                        {
                            "label": "Production",
                            "value": "prod",
                            "default": false
                        },
                        {
                            "label": "Test",
                            "value": "test",
                            "default": false
                        }
                    ]
                },
                {
                    "name": "Name",
                    "label": "Name",
                    "type": "string",
                    "immutable": true,
                    "hidden": false,
                    "required": false,
                    "secured": false,
                    "description": "Name provided for the service instance by the end user at time of
request",
                    "isSystemTag": true,
                    "customtype": "string",
                    "permission": "Read-Only",
                    "default": "${svc_instance.name}"
                },
                {
                    "name": "request_group",
                    "label": "request_group",
                    "type": "string",
                    "immutable": true,
                    "hidden": false,
                    "required": false,
                    "secured": false,
                    "description": "The current context group id of the current user that requested the
provisioning of the template. This is important because a user may be part of more than one group and this
identifies the context of which group the user made the request.",
                    "isSystemTag": true,
                    "customtype": "string",
                    "permission": "Read-Only",
                    "default": "${svc_instance.group}"
                },
                {
                    "name": "request_user",
                    "label": "request_user",
                    "type": "string",
                    "immutable": true,
                    "hidden": false,
                    "required": false,
                    "secured": false,
```

```
                    "description": "The user id of the current user that requested the provisioning of the
template.",
                    "isSystemTag": true,
                    "customtype": "string",
                    "permission": "Read-Only",
                    "default": "${svc_instance.owner}"
                },
                {
                    "name": "service_name",
                    "label": "service_name",
                    "type": "string",
                    "immutable": true,
                    "hidden": false,
                    "required": false,
                    "secured": false,
                    "description": "Name provided for the service instance by the end user at time of
request",
                    "isSystemTag": true,
                    "customtype": "string",
                    "permission": "Read-Only",
                    "default": "${svc_instance.name}"
                },
                {
                    "name": "service_identifier",
                    "label": "service_identifier",
                    "type": "string",
                    "immutable": true,
                    "hidden": false,
                    "required": false,
                    "secured": false,
                    "description": "Unique identifer generated by ICAM which is mapped to the service
instance of the template provisioned.",
                    "isSystemTag": true,
                    "customtype": "string",
                    "permission": "Read-Only",
                    "default": "${svc_instance.service_instance_id}"
                }
            ],
            "upgrade_from": [],
            "downgrade_to": [],
            "actions": [
                {
                    "type": "provision",
                    "name": "Provision",
                    "description": "Default action for deployment of service",
                    "input_parameters": [],
                    "flow": {
                        "conditions": [],
                        "templates": [],
                        "resthooks": [
                            {
                                "resthook": {
                                    "title": "resthook_cfb688",
                                    "id": "resthook00f1e7c5",
                                    "headers": {
                                        "accept": "application/json",
                                        "Content-Type": "application/json"
                                    },
                                    "editable": true,
                                    "url": "https://jsonplaceholder.typicode.com/todos/2",
                                    "method": "GET"
                                }
                            }
                        ],
                        "notifications": [],
                        "sequence": {
                            "0": "resthook00f1e7c5"
                        },
                        "error_sequence": {}
                    },
                    "output_parameters": [
                        {
                            "name": "todosUrl3",
                            "label": "todosUrl3",
                            "customtype": "string",
                            "type": "string",
                            "immutable": true,
                            "hidden": false,
                            "required": false,
                            "secured": false,
                            "default": "https://jsonplaceholder.typicode.com/todos/3"
```

```
                        }
                    ]
                }
            ],
            "plans": [
                {
                    "name": "Standard",
                    "description": "To deploy a Standard plan",
                    "actions": [],
                    "plan_parameters": []
                }
            ]
        }
    },
    "service_id": "621ccec6218004001ac4bc04",
    "tenantId": "cf5c3f7d-a70d-435c-9554-55d8e3441380",
    "namespaceId": "",
    "author": "admin",
    "specVersion": "v3",
    "status": "Published",
    "publishedTime": "2022-02-28T13:33:08.798Z",
    "tags": [
        {
            "name": "environment",
            "label": "Environment",
            "type": "string",
            "immutable": false,
            "hidden": false,
            "required": true,
            "secured": false,
            "description": "Unique identifer added to the provisioned VMs.",
            "isSystemTag": true,
            "customtype": "string",
            "permission": "Read-Write",
            "options": [
                {
                    "label": "Development",
                    "value": "dev",
                    "default": true
                },
                {
                    "label": "Quality Assurance",
                    "value": "qa",
                    "default": false
                },
                {
                    "label": "Production",
                    "value": "prod",
                    "default": false
                },
                {
                    "label": "Test",
                    "value": "test",
                    "default": false
                }
            ]
        },
        {
            "name": "Name",
            "label": "Name",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Name provided for the service instance by the end user at time of request",
            "isSystemTag": true,
            "customtype": "string",
            "permission": "Read-Only",
            "default": "${svc_instance.name}"
        },
        {
            "name": "request_group",
            "label": "request_group",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The current context group id of the current user that requested the provisioning
of the template. This is important because a user may be part of more than one group and this identifies the
```

```
context of which group the user made the request.",
            "isSystemTag": true,
            "customtype": "string",
            "permission": "Read-Only",
            "default": "${svc_instance.group}"
        },
        {
            "name": "request_user",
            "label": "request_user",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "The user id of the current user that requested the provisioning of the
template.",
            "isSystemTag": true,
            "customtype": "string",
            "permission": "Read-Only",
            "default": "${svc_instance.owner}"
        },
        {
            "name": "service_name",
            "label": "service_name",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Name provided for the service instance by the end user at time of request",
            "isSystemTag": true,
            "customtype": "string",
            "permission": "Read-Only",
            "default": "${svc_instance.name}"
        },
        {
            "name": "service_identifier",
            "label": "service_identifier",
            "type": "string",
            "immutable": true,
            "hidden": false,
            "required": false,
            "secured": false,
            "description": "Unique identifer generated by ICAM which is mapped to the service instance of the
template provisioned.",
            "isSystemTag": true,
            "customtype": "string",
            "permission": "Read-Only",
            "default": "${svc_instance.service_instance_id}"
        }
    ],
    "systemTags": true,
    "creationTime": "2022-02-28T13:31:50.023Z",
    "lastUpdated": "2022-02-28T13:33:08.798Z",
    "serviceInstanceCount": 0,
    "assignedCustomActions": [
        {
            "customActionName": "ind-action",
            "customActionId": "621cca7a218004001ac4bc03",
            "customInputParams": {
                "inputurl": "${svc_instance.output_parameters.todosUrl3}"
            }
        }
    ],
    "id": "621ccec6218004001ac4bc05",
    "actions": [
        {
            "type": "provision",
            "name": "Provision",
            "description": "Default action for deployment of service",
            "input_parameters": [],
            "flow": {
                "conditions": [],
                "templates": [],
                "resthooks": [
                    {
                        "resthook": {
                            "title": "resthook_cfb688",
                            "id": "resthook00f1e7c5",
                            "headers": {
                                "accept": "application/json",
```

```
                            "Content-Type": "application/json"
                        },
                        "editable": true,
                        "url": "https://jsonplaceholder.typicode.com/todos/2",
                        "method": "GET"
                    }
                }
            ],
            "notifications": [],
            "sequence": {
                "0": "resthook00f1e7c5"
            },
            "error_sequence": {}
        },
        "output_parameters": [
            {
                "name": "todosUrl3",
                "label": "todosUrl3",
                "customtype": "string",
                "type": "string",
                "immutable": true,
                "hidden": false,
                "required": false,
                "secured": false,
                "default": "https://jsonplaceholder.typicode.com/todos/3"
            }
        ]
    }
],
"plans": [
    {
        "name": "Standard",
        "description": "To deploy a Standard plan",
        "actions": [],
        "plan_parameters": []
    }
],
"tagId": "621c8329218004001ac4bbe2",
"warning": []
}
```

Additional information for the custom action to a service manifest API

1. You can follow either one of the steps to assign custom input parameters. In this case, **inputurl** values are assigned as shown:

   a. While assigning custom actions, you can attach the provison action output parameters of a service manifest to the custom action by using interpolation **${svc_instance.output_parameters.<o/p param>}**, which gets resolved when you try to deploy the independent custom action.

   Example

   ```
   {
       "custom_action_details": [
       {
           "customActionName": "ind-action",
           "customActionId": "<custom action id>",
           "customInputParams": {
           "inputurl": "${svc_instance.output_parameters.todosUrl3}"
           }
       }
     ]
   }
   ```

   b. You can directly assign the values which overrides the default values at the time of deployment of the custom action.

   Example

   ```
   {
       "custom_action_details": [
       {
           "customActionName": "ind-action",
           "customActionId": "<custom action id>",
           "customInputParams": {
           "inputurl": "http://example.com"
           }
       }
     ]
   }
   ```

2. You can assign custom actions for both publish and draft service manifest.

3. Reassigning custom actions for a service manifest overrides the previous assigned custom actions.

Unlock a custom action

You cannot unlock the custom action until it is unassigned with all the service versions associated with it.

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/CustomAction/<customid>/unlock?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d {}
```

Response


```
{
    "data": {
        "name": "ind-action",
        "ServiceEngine_InputFile": {
            "action": {
                "type": "custom",
                "name": "ind-action",
                "category": "demo-category",
                "description": " Independent custom action",
                "input_parameters": [
                    {
                        "name": "inputurl",
                        "label": "inputpurl",
                        "customtype": "string",
                        "type": "string",
                        "immutable": false,
                        "hidden": false,
                        "required": false,
                        "secured": false,
                        "isDynamic": false,
                        "default": "https://jsonplaceholder.typicode.com/todos/1"
                    }
                ],
                "flow": {
                    "conditions": [],
                    "templates": [],
                    "resthooks": [
                        {
                            "resthook": {
                                "title": "resthook_354d4a",
                                "id": "resthook98f44a65",
                                "headers": {
                                    "accept": "application/json",
                                    "Content-Type": "application/json"
                                },
                                "editable": true,
                                "url": "${input_parameters.inputurl}",
                                "method": "GET"
                            }
                        }
                    ],
                    "notifications": [],
                    "sequence": {
                        "0": "resthook98f44a65"
                    },
                    "error_sequence": {}
                },
                "output_parameters": [
                    {
                        "name": "resthookOutput",
                        "label": "resthookoutput",
                        "customtype": "string",
                        "type": "string",
                        "immutable": true,
                        "hidden": false,
                        "required": false,
                        "secured": false,
                        "description": "",
                        "default": "${resthooks.resthook98f44a65.output.body}"
                    }
                ]
            }
        },
        "categoryId": "621cc8e6218004001ac4bc00",
        "namespaceId": "cp4waiops",
```

```
        "tenantId": "cf5c3f7d-a70d-435c-9554-55d8e3441380",
        "creationTime": "2022-02-28T13:13:30.918Z",
        "lastUpdated": "2022-02-28T13:20:01.513Z",
        "status": "Unlocked",
        "id": "621cca7a218004001ac4bc03"
    }
}
```

Dynamic parameter discovery APIs

This section includes all dynamic parameter discovery related APIs.

- Clean Cache Data
- Update clear cache frequency

Clean Cache Data

You must be a cluster administrator or administrator to run this API. Otherwise, an unauthorized error message is displayed.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/dynamicparameterdiscovery/clean-cache?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Update clear cache frequency

You must be a cluster administrator or administrator to run this API. Otherwise, an unauthorized error message is displayed.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/dynamicparameterdiscovery/update-cache-config?
frequency=10&tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Application APIs

This section includes all application APIs with examples.

- Creating category for applications
- Creating an application
- Importing an application from GitHub repository
- Pushing an application to GitHub
- Duplicating an application
- Editing an application
- Deploying an application
- Publishing an application
- Retrieving application template

Creating category for applications

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppCategories?
ace_orgGuid=${MCM_TEAM}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d {"name": "AppCategorySample"}
```

Result

```
{
  {
    "name": "AppCategorySample",
    "id": "5fc9df7661b3e8001b7c2511",
    "tenantId": "83d50ee1-849b-4b33-bb25-9cb05f9bd87c"
}
```

Creating an application

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppModel/createApp?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @application_create.json
```

application_create.json

```
{
  "data": {
    "name": "ApplicationSample",
    "namespaceId": "abcdef123456",
      "spec": "apiVersion: app.k8s.io/v1beta1\nkind: Application\nmetadata:\n  labels:\n    app: nginx-app-
details\n  name: nginx-app-3\n  namespace: ns-sub-1\nspec:\n  componentKinds:\n  - group: app.ibm.com\n
kind: Subscription\n  - group: core.hybridapp.io\n    kind: Deployable\n  selector:\n    matchLabels:\n
app: nginx-app-details\nstatus: {}",
    "description": "",
    "categoryId": "5fc9df7661b3e8001b7c2511",
    "imagepath": "serviceicon_1.svg"
  }
}
```

Response:

```
{
    "data": {
        "name": "ApplicationSample",
        "spec": "apiVersion: app.k8s.io/v1beta1\nkind: Application\nmetadata:\n  labels:\n    app: nginx-app-
details\n  name: nginx-app-3\n  namespace: ns-sub-1\nspec:\n  componentKinds:\n  - group: app.ibm.com\n
kind: Subscription\n  - group: core.hybridapp.io\n    kind: Deployable\n  selector:\n    matchLabels:\n
app: nginx-app-details\nstatus: {}",
        "status": "Draft",
        "description": "",
        "longdescription": "",
        "namespaceId": "abcdef123456",
        "creationTime": "2020-12-04T08:06:00.395Z",
        "lastUpdated": "2020-12-04T08:06:00.395Z",
        "tenantId": "83d50ee1-849b-4b33-bb25-9cb05f9bd87c",
        "categoryId": "5fc9df7661b3e8001b7c2511",
        "inputparameters": [],
        "imagepath": "serviceicon_1.svg",
        "id": "5fc9ede861b3e8001b7c2515"
    }
}
```

Importing an application from GitHub repository

This API call imports an application from the GitHub repository.

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppModel/pullAppModelFromGIT?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @application_import.json
```

application_import.json

```
{
  "app_details": {
    "name": "Application",
    "categoryId": "5fc9df7661b3e8001b7c2511",
    "description": "ImportFromGitHub",
    "git_details": {
      "git_type": "github",
      "git_token": <github_token>,
      "file_path": "mysubdir/application.yaml",
      "git_ref": "master",
      "git_url": "http://mygithub/user/reponame"
    }
  }
}
```

Response:

```
{
  "pullAppModelFromGITResponse": {
    "statusCode": 200,
    "status": "success",
    "message": "Application imported successfully"
  }
}
```

Pushing an application to GitHub

This API pushes an application to GitHub.

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppModel/${APP_ID}/pushAppToGit?
ace_orgGuid=${MCM_TEAM}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @push_application.json
push_application.json
{
    "git_details": {
      "git_type": "github",
      "git_token": <github_token>,
      "file_path": "mysubdir/application.yaml",
      "git_ref": "master",
      "git_url": "http://mygithub/user/reponame"
    },
    "commit_details": {
      "commit_message": "Sample file to push an Application"
    }
}
```

Response:

```
{
    "pushAppToGITResponse": {
        "statusCode": 200,
        "status": "success",
        "message": "App pushed successfully"
    }
}
```

Duplicating an application

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppModel/${APP_ID}/duplicateApp?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d {"app_name": <Duplicate_APP_Name>}
```

Response:

```
{
    "data": {
        "name": <Duplicate_APP_Name>,
        "spec": "apiVersion: app.k8s.io/v1beta1\nkind: Application\nmetadata:\n  labels:\n    app: nginx-app-
details\n  name: nginx-app-3\n  namespace: ns-sub-1\nspec:\n  componentKinds:\n  - group: app.ibm.com\n
kind: Subscription\n  - group: core.hybridapp.io\n    kind: Deployable\n  selector:\n    matchLabels:\n
app: nginx-app-details\nstatus: {}",
        "status": "Draft",
        "description": "",
        "longdescription": "",
        "namespaceId": "abcdef123456",
        "creationTime": "2020-12-04T08:06:00.395Z",
        "lastUpdated": "2020-12-04T08:06:00.395Z",
        "tenantId": "83d50ee1-849b-4b33-bb25-9cb05f9bd87c",
        "categoryId": "5fc9df7661b3e8001b7c2511",
        "inputparameters": [],
        "imagepath": "serviceicon_1.svg",
        "id": "5fc9ede861b3e8001b7c2515"
    }
}
```

Editing an application

This API edits an application. Published applications cannot be edited.

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppModel/${APP_ID}/edit?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @application_edit.json
```

application_edit.json

```
{
  "data": {
    "name": "ApplicationSample",
    "spec": "apiVersion: app.k8s.io/v1beta1\nkind: Application\nmetadata:\n  labels:\n    app: nginx-app-
details\n  name: nginx-app-3\n  namespace: ns-sub-1\nspec:\n  componentKinds:\n  - group: app.ibm.com\n
kind: Subscription\n  - group: core.hybridapp.io\n    kind: Deployable\n  selector:\n    matchLabels:\n
app: ${matchLabel}\nstatus: {}",
    "status": "Draft",
    "description": "",
    "longdescription": "",
    "namespaceId": "",
    "creationTime": "2020-12-04T08:06:00.395Z",
    "lastUpdated": "2020-12-04T08:06:00.395Z",
    "tenantId": "83d50ee1-849b-4b33-bb25-9cb05f9bd87c",
    "categoryId": "5fc9df7661b3e8001b7c2511",
    "inputparameters": [
      {
        "name": "matchLabel",
        "label": "matchLable",
        "customtype": "string",
        "type": "string",
        "immutable": false,
        "hidden": false,
        "required": false,
        "secured": false,
        "description": "",
        "isDynamic": false,
        "permission": "Read-Write",
        "default": "nginx-app-details"
      }
    ],
    "imagepath": "serviceicon_1.svg",
    "id": <APP_ID>
  }
}
```

Response

```
{
    "data": {
        "name": "ApplicationSample",
        "spec": "apiVersion: app.k8s.io/v1beta1\nkind: Application\nmetadata:\n  labels:\n    app: nginx-app-
details\n  name: nginx-app-3\n  namespace: ns-sub-1\nspec:\n  componentKinds:\n  - group: app.ibm.com\n
kind: Subscription\n  - group: core.hybridapp.io\n    kind: Deployable\n  selector:\n    matchLabels:\n
app: ${matchLabel}\nstatus: {}",
        "status": "Draft",
        "description": "",
        "longdescription": "",
        "namespaceId": "",
        "creationTime": "2020-12-04T08:06:00.395Z",
        "lastUpdated": "2020-12-04T08:30:23.117Z",
        "tenantId": "83d50ee1-849b-4b33-bb25-9cb05f9bd87c",
        "categoryId": "5fc9df7661b3e8001b7c2511",
        "inputparameters": [
            {
                "name": "matchLabel",
                "label": "matchLable",
                "customtype": "string",
                "type": "string",
                "immutable": false,
                "hidden": false,
                "required": false,
                "secured": false,
                "description": "",
                "isDynamic": false,
                "permission": "Read-Write",
                "default": "nginx-app-details"
            }
        ],
```

```
        "imagepath": "serviceicon_1.svg",
        "id": <APP_ID>
    }
}
```

Deploying an application

This API deploys a published application.

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppModel/${APP_ID}/deploy?
ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d @application_deploy.json
```

application_deploy.json

```
{
  "name": "nginx-app-3",
  "parameters": [
    {
      "name": "matchLabel",
      "value": "nginx-app-details"
    }
  ]
}
```

Response:

```
{
    "instance": {
        "name": "nginx-app-3",
        "status": "Submitted",
        "creationTime": "2020-12-04T08:38:17.933Z",
        "lastUpdated": "2020-12-04T08:38:18.154Z",
        "stack_id": "5fc9f57af86820001b650685",
        "action": "CREATE",
        "logs": "Request queued",
        "namespaceId": "abcdef123456",
        "tenantId": "83d50ee1-849b-4b33-bb25-9cb05f9bd87c",
        "mode": "test",
        "id": "5fc9f57961b3e8001b7c2517",
        "appModelId": <APP_ID>
    }
}
```

Publishing an application

This API publishes an application.

```
curl -k -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppModel/${APP_ID}/publish?
ace_orgGuid=${MCM_TEAM}&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
```

Response:

```
{
    "data": {
        "name": "ApplicationSample",
        "spec": "apiVersion: app.k8s.io/v1beta1\nkind: Application\nmetadata:\n  labels:\n    app: nginx-app-
details\n  name: nginx-app-3\n  namespace: ns-sub-1\nspec:\n  componentKinds:\n  - group: app.ibm.com\n
kind: Subscription\n  - group: core.hybridapp.io\n    kind: Deployable\n  selector:\n    matchLabels:\n
app: ${matchLabel}\nstatus: {}",
        "status": "Published",
        "description": "",
        "longdescription": "",
        "namespaceId": "",
        "creationTime": "2020-12-04T08:06:00.395Z",
        "lastUpdated": "2020-12-04T08:43:42.368Z",
        "tenantId": "83d50ee1-849b-4b33-bb25-9cb05f9bd87c",
        "categoryId": "5fc9df7661b3e8001b7c2511",
        "inputparameters": [
            {
                "name": "matchLabel",
```

```
                "label": "matchLable",
                "customtype": "string",
                "type": "string",
                "immutable": false,
                "hidden": false,
                "required": false,
                "secured": false,
                "description": "",
                "isDynamic": false,
                "permission": "Read-Write",
                "default": "nginx-app-details"
            }
        ],
        "imagepath": "serviceicon_1.svg",
        "id": <APP_ID>
    }
}
```

Retrieving an application template

This API retrieves the application template details based on given application name.

```
curl -k -X GET \
'https://${TSA_HOSTNAME}/cam/composer/api/v1/AppModel?filter[where]
[name]=${APPNAME}&type=template&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_
ID}' \
  -H 'Authorization: Bearer ${TSA_BEARER_TOKEN}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache'
```

Response:

```
[
    {
        "description": "",
        "name": <APPNAME>,
        "type": "Application",
        "manifest": {
            "template_type": "AppModel",
            "template_format": "JSON",
            "template_provider": "Cloud Pak for Multicloud Management",
            "skip_formating": true,
            "template": {
                "templateVariables": {
                    "template_input_params": [
                        {
                            "name": "matchLabel",
                            "label": "matchLable",
                            "customtype": "string",
                            "type": "string",
                            "immutable": false,
                            "hidden": false,
                            "required": false,
                            "secured": false,
                            "description": "",
                            "isDynamic": false,
                            "permission": "Read-Write",
                            "default": "nginx-app-details"
                        }
                    ],
                    "template_output_params": []
                }
            }
        },
        "metadata": {
            "displayName": "ApplicationSample",
            "longDescription": ""
        },
        "id": <App_ID>,
        "default_template_version": "1.0",
        "template_version": {
            "id": <App_ID>,
            "ref": "1.0",
            "name": "1.0",
            "enabled": true,
            "compatible_versions": [
                "1.0"
            ],
            "displayUrl": ""
        },
```

```
        "template_versions": [
            {
                "id": <APP_ID>,
                "ref": "1.0",
                "name": "1.0",
                "enabled": true,
                "compatible_versions": [
                    "1.0"
                ]
            }
        ],
        "valid_connections": [],
        "provider_engine_versions": null
    }
]
```

Shared parameter APIs

This section includes all APIs for shared parameters.

- Create new Data Type
- List all Data Types
- Lookup Data Objects (implicit or explicit) for the Data Types specified in the request
- Resolve the Data Type variables for the template parameters
- Update Data Type
- Delete Data Type
- Create new Data Object
- Retrieve Data Object
- Update Data Object
- Delete Data Object

Create new Data Type

This API creates a new Data Type

```
curl -X POST \
  'https://{TSA_HOSTNAME}/cam/api/v1/datatypes?ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId=
{{TSA_TENANT_ID}}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  --insecure \
  -d '{
    "name": "DemoOneType",
    "label": "Demo One Type",
    "description": "Datatype for DemoOne",
    "attributes": [
        {
            "name": "hostip",
            "default": "0.0.0.0",
            "regex": "[0-9]+\\.[0-9]+\\.[0-9]+\\.[0-9]+",
            "type": "string",
            "required": "true",
            "secured": "false",
            "label": "Host IP address",
            "hidden": "false",
            "immutable": "false"
        },
        {
            "name": "port",
            "description": "Port number",
            "default": "80",
            "range": {
                "min": "80",
                "max": "9999"
            },
            "type": "counter",
            "required": "true",
            "secured": "false",
            "label": "Port",
            "hidden": "false",
            "immutable": "false"
        },
        {
```

```
            "name": "username",
            "default": "admin",
            "description": "User name",
            "hidden": false,
            "immutable": false,
            "label": "User Name",
            "required": true,
            "secured": false,
            "type": "string"
        },
        {
            "name": "password",
            "description": "Password for the host",
            "hidden": false,
            "immutable": false,
            "label": "Password",
            "regex": "[\\w]{8}",
            "required": true,
            "secured": true,
            "type": "password"
        }
    ]
}';
```

- **ROLE:**

  ADMINISTRATOR

- **Params:**

```
{
  name : [string],
  label: [string],
  description : [string],
  attributes : [
  {
    name: [string],
    default: [string],
    type: [string],
    label: [string],
    required: [boolean],
    hidden: [boolean],
    immutable: [boolean],
    secured: [boolean]
  }
 ]
}
```

- **Success Response:**
  - Code: `200` Content: `{ id : 1, name : "LDAPServer", description: "LDAP server data type", attributes: [...] }`
- **Error Response:**
  - Code: `400 BAD REQUEST` Content: `{ error : "Data type ${name} already defined." }`

  Or
  - **Code:** `401 UNAUTHORIZED` **Content:** `{ error : "You are unauthorized to make this request." }`

List all Data Types

This API lists all available Data Types.

```
curl -X GET \
  'https://{TSA_HOSTNAME}/cam/api/v1/datatypes?ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId=
{{TSA_TENANT_ID}}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  --insecure
```

- **ROLE:**

  ADMINISTRATOR

- **Success Response:**

- Code: `200` Content: `[{ id : 1, name : "LDAPServer", description: "LDAP server data type", attributes: [...] }, ...]`

Lookup Data Objects (implicit or explicit) for the Data Types specified in the request

```
curl -X POST \
  'https://{TSA_HOSTNAME}/cam/api/v1/datatypes/lookup?ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId=
{{TSA_TENANT_ID}}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d '{
  "datatypes" : [
    "DemoOneType"
  ]
}'
```

- **ROLE:**

  tenantMember

- **Params:**

  ```
  {
    datatypes : [datatype_name1, datatype_name2, ...]
  }
  ```

- **Success Response:**

  - Code: `200` Content:

    Example

    ```
    [
      {
        "name": "datatype_name1",
        "label": "datatype_label1",
        "dataobjects": [
        {
        "id": "1234567890",
        "name" : "dataobject name",
        "datatype": "datatype_name1",
        "provider": "IBM"
         }
       ]
      }
    ]
    ```

Resolve the Data Type variables for the template parameters

`POST /datatypes/resolve`

- **ROLE:**

  tenantMember

- **Params:**

  Example:

  ```
  {
     "input_dataobjects": [
     {
       "id": "1234567890",
       "name" : "dataobject name",
       "datatype": "datatype_name1",
       "provider": "IBM"
     }
    ],
     "parameters": {
     "template_input_params": [
     {
       "name": "param1",
       "type": "string",
       "default": "${ContentRuntime.output1}",
      }
     ]
    }
  }
  ```

- **Success Response:**
  - Code: `200` Content: `{ template_input_params: [{ name: 'param1', type: 'string', default: 'output1 value'}]}`

Update Data Type

```
curl -X PUT \
  'https://{TSA_HOSTNAME}/cam/api/v1/datatypes/{{DATATYPE_ID}}?
ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId={{TSA_TENANT_ID}}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d '{"description": "updated description"}'
```

- **ROLE:**

  ADMINISTRATOR

- **URL params:**

  datatype ID or datatype name

- **Params:**

  The following parameters are optional, for example, if you only want to update the 'attributes' the request body can contain only the 'attributes'.

  ```
  {
    name : [string],
    label: [string],
    description: [string],
    attributes:[
     {
      name: [string],
      default: [string],
      type: [string],
      label: [string],
      required: [boolean],
      hidden: [boolean],
      immutable: [boolean],
      secured: [boolean]
      }
    ]
  }
  ```

- **Success Response:**
  - Code: `200` Content: `{ id : 1, name : "LDAPServer", description: "LDAP server data type", attributes: [...] }`
- **Error Response:**
  - Code: `400 BAD REQUEST` Content: `{ error : "Data type ${name} already defined." }`

  Or
  - **Code:** `401 UNAUTHORIZED` **Content:** `{ error : "You are unauthorized to make this request." }`

Delete Data Type

```
curl -X DELETE \
  'https://{TSA_HOSTNAME}/cam/api/v1/datatypes/{{DATATYPE_ID}}?
ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId={{TSA_TENANT_ID}}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache'
```

- **ROLE:**

  ADMINISTRATOR

- **URL params:**

  datatype ID or datatype name

- **Success Response:**

- Code: `201` Content: None

- **Error Response:**
  - Code: `404 NOT FOUND` Content: `{ error : "Data type ${id} does not exist." }`

  Or
  - **Code:** `401 UNAUTHORIZED` **Content:** `{ error : "You are unauthorized to make this request." }`

Create new Data Object for specific namespace

This API creates a new Data Object.

```
curl -X POST \
  'https://{TSA_HOSTNAME}/cam/api/v1/dataobjects?ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId=
{{TSA_TENANT_ID}}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  --insecure \
  -d '{
  "datatype": "DemoOneType",
  "name": "DemoOneObject",
  "namespaceId": "{{NAMESPACE}}",
  "label": "DemoOneObject",
  "attributes": [
    {
      "name": "hostip",
      "value": "127.0.0.1"
    },
    {
      "name": "port",
      "value": 587
    },
    {
      "name": "username",
      "value": "user1"
    },
    {
      "name": "password",
      "value": "test1234"
    }
  ]
}'
```

- **ROLE:**

  ADMINISTRATOR

- **Success Response:**
  - Code: `200` Content: `{ id : 1, name : "LDAPServer.production", description: "My LDAP server for production", attributes:[...] }`
- **Error Response:**
  - Code: `400 BAD REQUEST` Content: `{ error : "Data object ${name} already defined." }`

  Or
  - **Code:** `401 UNAUTHORIZED` **Content:** `{ error : "You are unauthorized to make this request." }`

Optionally, you can add `cloudOE_spaceGuid=${NAMESPACE}` query parameter to create the Data Object in the namespace that `NAMESPACE` resolves to.

Retrieve Data Object

```
curl -X GET
'https://{TSA_HOSTNAME}/cam/api/v1/dataobjects?ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId=
{{TSA_TENANT_ID}}'
-H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}'
-H 'Content-Type: application/json'
-H 'cache-control: no-cache'
```

- **ROLE:**

  ADMINISTRATOR

- **URL params:**

dataobject ID

- **Success Response:**
  - Note: The attributes are merged with the datatype's attributes.
  - Code: `200` Content:

```
{
  "id" : 1,
  "name": "production",
  "label": "Production LDAP server",
  "datatype": "com.ibm.cam.ldapserver",
  description: "My LDAP server for production",
  attributes:[
  {
   "name": "ipaddress",
   "value": "127.0.0.1",
   "default": "",
   "type": "string",
   "required": "true",
   "secured": "false",
   "label": "IP address",
   "hidden": "false",
   "immutable": "false"
   },
   {
    "name": "port",
    "value": "367",
    "default": "",
    "type": "string",
    "required": "true",
    "secured": "false",
    "label": "Port",
    "hidden": "false",
    "immutable": "false"
    }
  ]
}
```

- **Error Response:**
  - Code: `400 BAD REQUEST` Content: `{ error : "Data object ${name} already defined." }`

Or

  - Code: `404 NOT FOUND` Content: `{ error : "Data object ${id} does not exist." }`

Or

  - **Code:** `401 UNAUTHORIZED` **Content:** `{ error : "You are unauthorized to make this request." }`

Optionally, you can add `cloudOE_spaceGuid=${NAMESPACE}` query parameter to create the Data Object in the namespace that `NAMESPACE` resolves to.

Update Data Object

```
curl -X PUT \
  'https://{TSA_HOSTNAME}/cam/api/v1/dataobjects/{{Dataobjects_ID}}?
ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId={{TSA_TENANT_ID}}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d '{"description": "updated description"}'
```

- **ROLE:**

  ADMINISTRATOR

- **URL params:**

  dataobject ID

- **Params:**

```
{
  name: [string],
  label: [string],
  datatype: [string],
  description: [string],
  attributes:[
```

```
    {
     // attribute definition
     ...
    }
   ]
}
```

- **Success Response:**
  - Code: `200` Content: `{ id : 1, name : "LDAPServer.production", description: "My LDAP server for production", attributes:[...] }`
- **Error Response:**
  - Code: `400 BAD REQUEST` Content: `{ error : "Data object ${name} already defined." }`

  Or
  - Code: `404 NOT FOUND` Content: `{ error : "Data object ${id} does not exist." }`

  Or
  - **Code:** `401 UNAUTHORIZED` **Content:** `{ error : "You are unauthorized to make this request." }`

Optionally, you can pass `cloudOE_spaceGuid=${NAMESPACE}` in the query parameter to change the namespace of the Data Object. The modified namespace value is the value that `NAMESPACE` resolves to.

Delete Data Object

```
curl -X DELETE \
  'https://{TSA_HOSTNAME}/cam/api/v1/dataobjects/{{Dataobjects_ID}}?
ace_orgGuid=all&cloudOE_spaceGuid=default&tenantId={{TSA_TENANT_ID}}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache'
```

- **ROLE:**

  ADMINISTRATOR
- **URL params:**

  dataobject ID
- **Success Response:**
  - Code: `200` Content: `{ id : 1, name : "LDAPServer.production", description: "My LDAP server for production", attributes:[...] }`
- **Error Response:**
  - Code: `404 NOT FOUND` Content: `{ error : "Data object ${id} does not exist." }`

  Or
  - **Code:** `401 UNAUTHORIZED` **Content:** `{ error : "You are unauthorized to make this request." }`

SMTP Connections APIs

Learn more about SMTP connection APIs.

- Create SMTP connection
- Modify SMTP connection
- Test SMTP connection
- Get list of all SMTP connections
- Get list of SMTP connections by ID
- Delete SMTP connection

Create SMTP connection

To use email notifications, you must configure a connection to the SMTP server by running this API. Creating a secondary connection is optional.

1. Create a primary connection.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create_primary.json \
"https://${TSA_HOSTNAME}/cam/composer/api/v1/smtpconnections?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}
```

POST method

Example request body:

cc_create_primary.json for primary connection:

```
{
   "name": "PRIMARY",
   "connection_parameters": [{
       "name": "hostname",
       "value": "smtp.example.com"
   }, {
       "name": "port",
       "value": 25
   }, {
       "name": "username",
       "value": "myusername"
   }, {
       "name": "password",
       "value": "mypassword"
 }]
}
```

2. Create a secondary connection.

```
curl -k -X POST \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
-H "Content-Type: application/json" \
-H "Accept: application/json" \
-d @cc_create_secondary.json \
"https://${TSA_HOSTNAME}/cam/composer/api/v1/smtpconnections?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}
```

POST method

Example request body: cc_create_secondary.json for secondary connection

```
{
    "name": "SECONDARY",
    "connection_parameters": [{
           "name": "hostname",
           "value": "test"
    }, {
           "name": "port",
           "value": 25
    }, {
           "name": "username",
           "value": "test"
    }, {
           "name": "password",
           "value": "test"
    }]
}
```

The notification tries to connect to the SMTP server defined in the primary connection. If this connection fails, the notification tries to connect to the SMTP server defined in the secondary connection.

Modify SMTP connection

```
curl -X POST \
 'https://${TSA_HOSTNAME}/cam/composer/api/v1/smtpconnections/patchConnectionParametersById?
ace_orgGuid=all&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
  -H "Authorization: Bearer ${TSA_BEARER_TOKEN}"\
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d '{
        "id":"{{SMTPCONNECTION_ID}}",
        "connection_parameters":[
           {"name":"port","value": 587},
           {"name":"password","value":"my-new-password"}
```

```
        ]
    }'
```

Where **TSA_TENANT_ID** cannot be null.

POST method

The **patchConnectionParameterById** is used to update some of the parameters in the existing SMTP Connection record. Both **patchConnectionParameterById** and **connection_parameters** are mandatory. **TSA_TENANT_ID** cannot be null.

Test SMTP connection

Run this API to test SMTP connection parameters before you create or save a connection.

```
curl -X POST \
  'https://${TSA_HOSTNAME}/cam/composer/api/v1/smtpconnections/testEmail?
ace_orgGuid=all&tenantId=${TSA_TENANT_ID}' \
  -H 'Authorization: Bearer {{TSA_BEARER_TOKEN}}' \
  -H 'Content-Type: application/json' \
  -H 'cache-control: no-cache' \
  -d '{
    "to":"{{SEND-TEST-EMAIL-TO}}",
    "conn_params":[
        {"name":"hostname","value":"{{SMTP_HOST}}"},
        {"name":"port","value": "{{SMTP_PORT}}" },
        {"name":"username","value":"{{USERNAME}}"},
        {"name":"password","value":"{{PASSWORD}}"}
    ]
}'
```

Where **TSA_TENANT_ID** cannot be null.

POST method

The **/testEmailByID** is similar to **/testEmail** but it uses persisted parameters from a particular record. Optionally, you can send **conn_params** argument whenever you want to use some parameters with changed values.

POST method

```
curl -X POST \
    'https://${TSA_HOSTNAME}/cam/composer/api/v1/smtpconnections/testEmailById?
ace_orgGuid=all&cloudOE_spaceGuid=${NAMESPACE}&tenantId=${TSA_TENANT_ID}' \
    -H "Authorization: Bearer ${TSA_BEARER_TOKEN}"\
    -H 'Content-Type: application/json' \
    -H 'cache-control: no-cache' \
    -d '{
        "id":"{{SMTPCONNECTION_ID}}",
        "to":"my-test-email@test.xyz.net",
        "conn_params":[
           {"name":"port","value": 587},
           {"name":"password","value":"my-new-password"}
      ]
     }'
```

Where,

- The **conn_params** is optional argument. It can contain subset of parameters that you edit in the user interface. The rest of the parameters are taken from SMTPConnection for the id.
- The **id** and **to** values are mandatory arguments.
- **TSA_TENANT_ID** cannot be null.

Get list of all SMTP connections

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}"\
 "https://${TSA_HOSTNAME}/cam/composer/api/v1/smtpconnections?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}
```

GET method

Get list of SMTP connections by ID

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
```

```
"https://${TSA_HOSTNAME}/cam/composer/api/v1/smtpconnections/<connection-id>?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}
```

GET method

Delete SMTP connection

```
curl -k -X DELETE \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}"\
 "https://${TSA_HOSTNAME}/cam/composer/api/v1/smtpconnections/<connection-id>?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}
```

DELETE Method

Usage specific APIs

As a cluster administrator, you can run the usage specific APIs.

- Retrieve the current number of deployed virtual machines
- Retrieve the maximum number of virtual machines for each month
- Download the managed virtual machines report

You must be a cluster administrator to run these APIs. Otherwise, an unauthorized error message is displayed.

Retrieve the current number of deployed virtual machines

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/managedvirtualmachines/countActive?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Retrieve the maximum number of virtual machines for each month

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/managedvirtualmachines/highwatermark?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Download the managed virtual machines report

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/managedvirtualmachines/download?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}"
```

Example to retrieve the list of deployed templates

Example API flow to retrieve the list of deployed templates.

A series of API calls to get the list of deployed templates:

```
curl -k -X POST \
-H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8" \
-d "grant_type=password&username=${TSA_USER}&password=${TSA_USER_PASSWORD}&scope=openid"\
https://${MCM_CLUSTER_HOSTNAME}/idprovider/v1/auth/identitytoken
```

Example:

```
curl -k -X POST -H "Content-Type: application/x-www-form-urlencoded;charset=UTF-8" -d
"grant_type=password&username=admin&password=admin&scope=openid"
https://<MCM_CLUSTER_HOSTNAME>/idprovider/v1/auth/identitytoken
```

Response:

```
{
"access_token":"gvWTQGcb9DzZZb5Oj6GYRJm0OOUBYTCxgjJUlI0dDURsw6p2uWdG3jDCe8S8UCqLVSoU3wwr9toWKGtISkcMrttBBGzET
8hDHbLd3DXUlPMt1jNZAX65aRLvEhLHouVc1HlWQVavVYT9RBeHfNcTkCbYsaToh8rWmzwxAQsN5vRESk727VeTapSfHstx0CL2oUNanHuu73
```

```
hukWQ
I3MEsN2X9q1gnGIyUdeSdHQZYJ1Gll8h0GV9vNzA48dgoEPSPtbGK3vRt3p2gK4lzzzwjjPiRX8LAqPXYxjc3GN4RENkhF2XhHLsZJqWAni9V
iRhjQiRTOZjpTLu8xdYg4HPwxqywBviIhUVPyqDe2CaHqjl2JVzeVoodByY7rsSEXiZviwjrvD6vcoQ7nLLkQ4JrpdxZP7Zm1SKluNQ0I7Zol
Gz1gk81
Y1F8OKISZZSEyp7PZnHKI6RwNiFtWc54xvAh8tlQ15NHlF3ZsyTOa2lIKFJeO1SW8ejMLQaUxfS5uVIgHnOetY4auLU7X82IaR9r6n2BB9ftU
Wq7ZKf2uG1pC1MwW5ployNvu1sCrgh7bvsthwGRlOdRp04zE94rmidihnCGCUlcf6hWxKNq8cqVsLmk5IJKfpbLVxu2OYFWEMd0bqL1YzaDpd
gtDD5iH
0BNw83d2omjisukkfINNA0T0Z7vpuxuXZ13NBC7fsK1FvN1WWDDs7oHcZ8SPYsITu4cbdjuVayilE1QCAxTzyDVz70DQa84qv1sqczn5jPGVh
5cyBVikTEPd6BEQuBhV7Tbq2WVhuS3IuAHWUqfQb085Xb14SqQOXRtatlcRZb11npg44WLXTqH4HliglpyQpbAJTWGCV5AgDWc8fGjjUIz6nW
o5O70Omr
cvNOhVKzshXv49PqaONdLQpQ13LnUrRCsD9L3IpdJfxojR6eCSG4qQW1JVJC8uSQtjKYkqvdlmAPrxUgMv8yrPfIEOH1tV6h45SricSYKoOfB
G72DqIXH","token_type":"Bearer","expires_in":43199,"scope":"openid","refresh_token":"fvNZN6AjYbS3RDXyi0rchBpb
cezeMCJT5
BmPEEhOmC529rr0Bx","id_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiMnhka3hoaDY5OTdpdHE2bmFsZ
HMiLCJyZWFsbU5hbWUiOiJjdXN0b21SZWFsbSIsInVuaXF1ZVNlY3VyaXR5TmFtZSI6ImFkbWluIiwiaXNzIjoiaHR0cHM6Ly9teWNsdXN0ZX
IuaWNwOjk
0NDMvb2lkYy9lbmRwb2ludC9PUCIsImF1ZCI6ImMzNGJlMjVjNTQ1MDlmMDA3MjRiYjAxOTI2MGZhNzJlIiwiZXhwIjoxNTA5NzMxMjU5LCJp
YXQiOjE1MDk3MzEyNTksInN1YiI6ImFkbWluIn0.tp7MKObvsAqzdARWf3U24Us_f3ZTdJGWl2IC2vYUisUTwKrhC7thoPSFViUmbpOhbswtY
XNuGxR2E6E
KsnaU7TDQhrbZdLWWB2E8OC-Kzd6ITEeCmsV8RrWFQmPm5uidCTBmkbSoImcQ2I6dqtD66xiWc0Gq-
sWceVrDL63VpuUbQxuJfmmCswEcDVybpJBOM2xHvGkhkLxwU9pRtnvtTbC1BoMNsIzZn4gR3vpm1ViIMS6_FVvFuzAHnZ_95fYZ1RTVXr5QIm
uN4vC-JIZb8KCxLqHvM9GUHus_XaQNq3npBRXpLluL
YI29yX4VkQNtbS_PKWao1HKGcFNSqLRS8w"
}
```

Note: Save the `access_token` from the response and use it as a bearer token in following API calls.

```
export
TSA_BEARER_TOKEN=gvWTQGcb9DzZZb5Oj6GYRJm0OOUBYTCxgjJUlI0dDURsw6p2uWdG3jDCe8S8UCqLVSoU3wwr9toWKGtISkcMrttBBGzE
T8hDHbLd3DXUlPMt1jNZAX65aRLvEhLHouVc1HlWQVavVYT9RBeHfNcTkCbYsaToh8rWmzwxAQsN5vRESk727VeTapSfHstx0CL2oUNanHuu7
3hu
kWQI3MEsN2X9q1gnGIyUdeSdHQZYJ1Gll8h0GV9vNzA48dgoEPSPtbGK3vRt3p2gK4lzzzwjjPiRX8LAqPXYxjc3GN4RENkhF2XhHLsZJqWAn
i9ViRhjQiRTOZjpTLu8xdYg4HPwxqywBviIhUVPyqDe2CaHqjl2JVzeVoodByY7rsSEXiZviwjrvD6vcoQ7nLLkQ4JrpdxZP7Zm1SKluNQ0I7
ZolGz1gk81
Y1F8OKISZZSEyp7PZnHKI6RwNiFtWc54xvAh8tlQ15NHlF3ZsyTOa2lIKFJeO1SW8ejMLQaUxfS5uVIgHnOetY4auLU7X82IaR9r6n2BB9ftU
Wq7ZKf2uG1pC1MwW5ployNvu1sCrgh7bvsthwGRlOdRp04zE94rmidihnCGCUlcf6hWxKNq8cqVsLmk5IJKfpbLVxu2OYFWEMd0bqL1YzaDpd
gtDD5iH0BN
w83d2omjisukkfINNA0T0Z7vpuxuXZ13NBC7fsK1FvN1WWDDs7oHcZ8SPYsITu4cbdjuVayilE1QCAxTzyDVz70DQa84qv1sqczn5jPGVh5cy
BVikTEPd6BEQuBhV7Tbq2WVhuS3IuAHWUqfQb085Xb14SqQOXRtatlcRZb11npg44WLXTqH4HliglpyQpbAJTWGCV5AgDWc8fGjjUIz6nWo5O
70OmrcvNOh
VKzshXv49PqaONdLQpQ13LnUrRCsD9L3IpdJfxojR6eCSG4qQW1JVJC8uSQtjKYkqvdlmAPrxUgMv8yrPfIEOH1tV6h45SricSYKoOfBG72DqIXH
```

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
https://${TSA_HOSTNAME}/cam/tenant/api/v1/tenants/getTenantOnPrem
```

Example:

```
curl -k -X GET -H "Authorization: Bearer
gvWTQGcb9DzZZb5Oj6GYRJm0OOUBYTCxgjJUlI0dDURsw6p2uWdG3jDCe8S8UCqLVSoU3wwr9toWKGtISkcMrttBBGzET8hDHbLd3DXUlPMt1
jNZAX65aRLvEhLHouVc1HlWQVavVYT9RBeHfNcTkCbYsaToh8rWmzwxAQsN5vRESk727VeTapSfHs
tx0CL2oUNanHuu73hukWQI3MEsN2X9q1gnGIyUdeSdHQZYJ1Gll8h0GV9vNzA48dgoEPSPtbGK3vRt3p2gK4lzzzwjjPiRX8LAqPXYxjc3GN4
RENkhF2XhHLsZJqWAni9ViRhjQiRTOZjpTLu8xdYg4HPwxqywBviIhUVPyqDe2CaHqjl2JVzeVoodByY7rsSEXiZviwjrvD6vcoQ7nLLkQ4Jr
pdxZP7Zm1
SKluNQ0I7ZolGz1gk81Y1F8OKISZZSEyp7PZnHKI6RwNiFtWc54xvAh8tlQ15NHlF3ZsyTOa2lIKFJeO1SW8ejMLQaUxfS5uVIgHnOetY4auL
U7X82IaR9r6n2BB9ftUWq7ZKf2uG1pC1MwW5ployNvu1sCrgh7bvsthwGRlOdRp04zE94rmidihnCGCUlcf6hWxKNq8cqVsLmk5IJKfpbLVxu
2OYFWEMd0b
qL1YzaDpdgtDD5iH0BNw83d2omjisukkfINNA0T0Z7vpuxuXZ13NBC7fsK1FvN1WWDDs7oHcZ8SPYsITu4cbdjuVayilE1QCAxTzyDVz70DQa
84qv1sqczn5jPGVh5cyBVikTEPd6BEQuBhV7Tbq2WVhuS3IuAHWUqfQb085Xb14SqQOXRtatlcRZb11npg44WLXTqH4HliglpyQpbAJTWGCV5
AgDWc8fGjj
UIz6nWo5O70OmrcvNOhVKzshXv49PqaONdLQpQ13LnUrRCsD9L3IpdJfxojR6eCSG4qQW1JVJC8uSQtjKYkqvdlmAPrxUgMv8yrPfIEOH1tV6
h45SricSYKoOfBG72DqIXH" https://9.37.194.250/cam/tenant/api/v1/tenants/getTenantOnPrem
```

Response:

```
{"id":"5c0e0a5e-d431-468c-9a11-a3df3978e424","name":"ICP
Account","ownerId":"On_Prem","wlpClientId":"a7lLLVER8bTekvyNmJjEdWY6XO7NGb3I","namespaces":
[{"name":"default","uid":"default"},{"name":"kube-public","uid":"kube-public"},{"name":"kube-
system","uid":"kube-system"},{"name":"services","uid":"services"}]}
tenantId = response.id = 5c0e0a5e-d431-468c-9a11-a3df3978e424
export TSA_TENANT_ID=5c0e0a5e-d431-468c-9a11-a3df3978e424
MCM team (ace_orgGuid) = all
export MCM_TEAM=all
namespace (cloudOE_spaceGuid) = response.namespaces[0].uid = default
export NAMESPACE=default
```

```
curl -k -X GET \
-H "Authorization: Bearer ${TSA_BEARER_TOKEN}" \
"https://${TSA_HOSTNAME}/cam/api/v1/stacks?
tenantId=${TSA_TENANT_ID}&ace_orgGuid=${MCM_TEAM}&cloudOE_spaceGuid=${NAMESPACE}"
```

Example:

```
curl -k -X GET -H "Authorization: Bearer
gvWTQGcb9DzZZb5Oj6GYRJm0OOUBYTCxgjJUlI0dDURsw6p2uWdG3jDCe8S8UCqLVSoU3wwr9toWKGtISkcMrttBBGzET8hDHbLd3DXUlPMt1
jNZAX65aRLvEhLHouVc1HlWQVavVYT9RBeHfNcTkCbYsaToh8rWmzwxAQsN5vRESk727VeTapSfHstx0CL2oUNanH
uu73hukWQI3MEsN2X9q1gnGIyUdeSdHQZYJ1Gll8h0GV9vNzA48dgoEPSPtbGK3vRt3p2gK4lzzzwjjPiRX8LAqPXYxjc3GN4RENkhF2XhHLs
ZJqWAni9ViRhjQiRTOZjpTLu8xdYg4HPwxqywBviIhUVPyqDe2CaHqjl2JVzeVoodByY7rsSEXiZviwjrvD6vcoQ7nLLkQ4JrpdxZP7Zm1SKl
uNQ0I7ZolGz1gk81Y1F8OK
ISZZSEyp7PZnHKI6RwNiFtWc54xvAh8tlQ15NHlF3ZsyTOa2lIKFJeO1SW8ejMLQaUxfS5uVIgHnOetY4auLU7X82IaR9r6n2BB9ftUWq7ZKf
2uG1pC1MwW5ployNvu1sCrgh7bvsthwGRlOdRp04zE94rmidihnCGCUlcf6hWxKNq8cqVsLmk5IJKfpbLVxu2OYFWEMd0bqL1YzaDpdgtDD5i
H0BNw83d2omjisukkfINNA
0T0Z7vpuxuXZ13NBC7fsK1FvN1WWDDs7oHcZ8SPYsITu4cbdjuVayilE1QCAxTzyDVz70DQa84qv1sqczn5jPGVh5cyBVikTEPd6BEQuBhV7T
bq2WVhuS3IuAHWUqfQb085Xb14SqQOXRtatlcRZb11npg44WLXTqH4HliglpyQpbAJTWGCV5AgDWc8fGjjUIz6nWo5O70OmrcvNOhVKzshXv4
9PqaONdLQpQ13LnUrRCsD9L
3IpdJfxojR6eCSG4qQW1JVJC8uSQtjKYkqvdlmAPrxUgMv8yrPfIEOH1tV6h45SricSYKoOfBG72DqIXH"
https://9.30.37.11/cam/api/v1/stacks?tenantId=5c0e0a5e-d431-468c-9a11-
a3df3978e424&ace_orgGuid=all&cloudOE_spaceGuid=default
```

Response:

```
[{"name":"MyvSphereDeploy","created_at":"2017-11-
02T18:08:00.818Z","templateId":"59f8b7b50377070020bdf501","namespaceId":"default","templateName":"SingleVirtu
alMachine-Hugh","templateCategory":"userCreated","templateProvider":"VMware
vSphere","template_type":"Terraform","template_format":"HCL","applied_state_id":"59fb5f01873a17001f1d3896","l
atest_state_id":"59fb5f01873a17001f1d3896","cloud_connection_ids":
["59f8b79e0377070020bdf4fe"],"id":"59fb5f00873a17001f1d3895","tenantId":"5c0e0a5e-d431-468c-9a11-
a3df3978e424","status":"IN_PROGRESS","message":"","action":"CREATE","stack_id":"c36cc6d0-bff8-11e7-ac54-
5bb358fd234d","applied_status":"IN_PROGRESS","applied_action":"CREATE"}]
```

Sample Python script

This section has a sample python script for reference.

```
import requests
import json

print("Managed services APIs - Demo Python Script")

protocol='https://'
tsaHostname='tsa_hostname'
mcmClusterHostname='MCM_CLUSTER_HOSTNAME'
user='TSA_admin_id'
pw='TSA_admin_password'

tsaEndpoint=protocol + tsaHostname
mcmEndpoint=protocol + mcmClusterHostname

<!--GET access_token from IBM Cloud Pak for AIOps-->

postdata = "grant_type=password&username=" + user + "&password=" + pw + "&scope=openid"
head = {'Content-Type':'application/x-www-form-urlencoded', 'Accept':'application/json', "charset" : "UTF-8"}
ret = requests.post(mcmEndpoint + "/idprovider/v1/auth/identitytoken",
                    data=postdata,
                    headers=head,
                    verify=False)
access_token=ret.json()["access_token"]
print("access_token", access_token)

<!-- GET tenantId, team and namespace from Managed services-->

head = {"Authorization": "bearer " + access_token}
ret = requests.get(tsaEndpoint + "/cam/tenant/api/v1/tenants/getTenantOnPrem",
                    headers=head,
                    verify=False)
tenantId=ret.json()["id"]
namespace=ret.json()["namespaces"][0]["uid"]
if user != 'admin':
    team=ret.json()["namespaces"][0]["teamId"]
else:
    team='all'
print("tenantId", tenantId)
print("team", team)
print("namespace", namespace)

<!-- GET cloudconnections from Managed services-->

parameters={"tenantId":tenantId, "ace_orgGuid":team}
```

```
head = {"Authorization": "bearer " + access_token, 'Accept':'application/json'}
ret = requests.get(tsaEndpoint + "/cam/api/v1/cloudconnections",
                     headers=head,
                     params=parameters,
                     verify=False)
cloudConnections=json.dumps(ret.json())
print("cloudConnections", cloudConnections)

<!-- GET stacks from Managed services-->

parameters={"tenantId":tenantId, "ace_orgGuid":team, "cloudOE_spaceGuid":namespace}
head = {"Authorization": "bearer " + access_token, 'Accept':'application/json'}
ret = requests.get(tsaEndpoint + "/cam/api/v1/stacks",
                     headers=head,
                     params=parameters,
                     verify=False)
stacks=json.dumps(ret.json())
print("stacks", stacks)
<!-- GET templates from Managed services -->
parameters={"tenantId":tenantId, "ace_orgGuid":team}
head = {"Authorization": "bearer " + access_token, 'Accept':'application/json'}
ret = requests.get(tsaEndpoint + "/cam/api/v1/templates",
                     headers=head,
                     params=parameters,
                     verify=False)
templates=json.dumps(ret.json())
print("templates", templates)
<!-- GET templates from Managed services with name filter-->
parameters={"tenantId":tenantId, "ace_orgGuid":team, "filter":'{"where": {"name": "SingleVirtualMachine" }}'}
head = {"Authorization": "bearer " + access_token, 'Accept':'application/json'}
ret = requests.get(tsaEndpoint + "/cam/api/v1/templates",
                     headers=head,
                     params=parameters,
                     verify=False)
templates=json.dumps(ret.json())
templateId=ret.json()[0]["id"]
print("templates", templates)
print("templateId", templateId)
<!-- GET sharedruntimeinstances from Managed services-->
parameters={"tenantId":tenantId, "ace_orgGuid":team, "cloudOE_spaceGuid":namespace}
head = {"Authorization": "bearer " + access_token, 'Accept':'application/json'}
ret = requests.get(tsaEndpoint + "/cam/api/v1/sharedruntimeinstances",
                     headers=head,
                     params=parameters,
                     verify=False)
sharedruntimeinstances=json.dumps(ret.json())
print("sharedruntimeinstances", sharedruntimeinstances)
<!-- GET order from Managed services-->
parameters={"tenantId":tenantId, "ace_orgGuid":team, "cloudOE_spaceGuid":namespace}
head = {"Authorization": "bearer " + access_token, 'Accept':'application/json'}
ret = requests.get(tsaEndpoint + "/cam/composer/api/v1/ServiceInstances",
                     headers=head,
                     params=parameters,
                     verify=False)
order=json.dumps(ret.json())
print("order", order)
```

Managed services Content APIs

The Managed services Content APIs manage cookbooks and roles in the pattern manager.

The Managed services Content APIs are used to list, update, and delete content on the pattern manager. The pattern manager interacts with the Chef server and software repository to manage cookbooks, recipes, and roles in your Content Runtime environment. When deploying a content runtime, you can install a new Chef server or bring your own; the pattern manager will interact with either. Managed services has a set of middleware cookbooks that can be loaded automatically when deploying the Content Runtime environment or can be loaded manually after the Content Runtime has been created. Load cookbooks, recipes, and roles from IBM, from your own repositories, or from Chef Supermarket.

Access to these APIs is granted with either the access token or the admin token. The access and admin tokens were specified when deploying the Content Runtime. Any API that accepts the access token also accepts the admin token. When using these APIs, the access or admin token must be specified with an authorization HTTP header. For example,

```
# curl -H "Authorization: Bearer $ACCESS_TOKEN" https://pattern_manager:5443/v1/info/chef
```

Image management API data

API Version

1.0.0

API URI components

Scheme

HTTPS

Port number

5443

Command output format

application/json

Upload cookbooks and roles

Uploads cookbooks from a git repository, chef supermarket, or local repository to the pattern manager.

`POST /v1/upload/chef`

Description

Uploads cookbooks from a git repository, Chef supermarket, or local repository to the pattern manager. Valid sources are the chef supermarket, github.com, gitlab.com, and local repository on the pattern manager file system. To ensure the latest updates are available on your pattern manager, call this API every time there is an update to the cookbooks and roles in your repository.

The pattern manager access token is required to perform this request.

Directory Structure

For cookbooks that are loaded from a git or local repository, it is assumed that they are in a folder named cookbooks. This cookbooks folder can either be at the root directory of your repository or in a folder named Chef that is at the root directory of your repository. Similarly, it is assumed that roles are in a folder named roles, which is either at the root directory of your repository or in a folder named chef within the root directory of your repository.

Directory Structure Example

The following code block depicts an example directory structure:

```
.
|-- github.ibm.com
    |-- example_org
        |-- example_repository
            |-- chef
                |-- cookbooks
                    |-- README.md
                    |-- attributes.json
                    |-- components.json
                    |-- cookbook_metadata.json
                    |-- metadata.rb
                    |-- recipes.json
                |-- roles
                    |-- create_user.json
                    |-- database_v10_install.json
                    |-- database_v15_install.json
```

Parameters

The following table lists the parameters that are used to upload cookbooks and roles to the pattern manager:

Type	Name	Description	Schema	Default	Required
Body	authorization	JSON object containing a personal_access_token property	JSON object		Yes for git repository
Body	cookbook_name	Name of the cookbook to load when uploading from the chef supermarket. Only valid when setting source_repos to chef_supermarket	String		Yes, when using chef supermarket

Type	Name	Description	Schema	Default	Required
Body	cookbook_version	Version of the cookbook to load when uploading from the chef supermarket. Only valid when setting source_repos to chef_supermarket	String		No
Body	deps	JSON array of dependencies required for the cookbook being installed from the chef supermarket. Each item must contain the cookbook_name and an optional cookbook_version parameter. Only valid when setting source_repos to chef_supermarket	JSON array		No
Body	personal_access_token	GitHub personal access token	String		Yes for git repository
Body	cookbooks	States whether cookbooks should be uploaded	Boolean	True	No
Body	org	Organization where your repositories reside. For GitLab, this is your group name	String		Yes for git repository
Body	branch	GitHub branch for your repositories	String	master	No
Body	repos	For git or local repositories, the regular expression to filter repositories (GitHub) or projects (GitLab) where you want to search for cookbooks and roles	String	.*	No
Body	repos_hostname	The hostname or IP address of your git repository if it does not match the value in source_repos	String	Value of source_repos	No
Body	repos_protocol	The protocol (http or https) used by your git repository	String	https	No
Body	roles	States whether roles should be uploaded	Boolean	False when source_repos is set to chef_supermarket, True otherwise	No
Body	source_repos	Hostname of the git repository. Valid options include chef_supermarket, github.com, gitlab.com, or a path to your local repository in file URI scheme.	String		Yes

Example

The following example shows the request that would be used to import all cookbooks and roles from the preceding example directory structure.

```
POST /v1/upload/chef

Header: Authorization = Bearer $ACCESS_TOKEN

{
  "authorization": {
    "personal_access_token": "38eab5732f7411bedb976afcc90a25432d798bda"
  },
  "cookbooks": "True",
  "source_repos": "github.com",
  "org": "example_org",
  "repos": "example_.*",
  "roles": "True"
}
```

This example shows a request to import from a GitLab repository that is hosted somewhere other than the gitlab.com. The value specified for org is the GitLab group name and the repos regular expression will filter projects belonging to that group.

```
POST /v1/upload/chef

Header: Authorization = Bearer $ACCESS_TOKEN

{
  "authorization": {
    "personal_access_token": "7411bedb9738eab5732f6afc32d798bdac90a254"
  },
  "cookbooks": "True",
  "source_repos": "gitlab.com",
  "org": "production_group",
  "repos": "cookbooks_.*",
  "roles": "True",
  "repos_hostname": "gitlab-pc.ibm.com",
  "protocol": "http"
}
```

This example shows a request to import the Java cookbook from the chef supermarket. The Java cookbook is dependent on the four other cookbooks listed in the **deps** parameter. If a cookbook in the **dep** array already exists on the Chef server and the **overwrite_existing** parameter is set to true, then it gets overwritten.

```
POST /v1/upload/chef

Header: Authorization = Bearer $ACCESS_TOKEN

{
  "cookbook_name": "java",
  "cookbook_version": "1.50.0",
  "source_repos": "chef_supermarket",
  "deps": [
    {
      "cookbook_name": "apt",
      "cookbook_version": "6.1.4"
    },
    {
      "cookbook_name": "homebrew",
      "overwrite_existing": "true"
    },
    {
      "cookbook_name": "windows"
    },
    {
      "cookbook_name": "ohai"
    }
  ]
}
```

This example shows a request to import cookbooks and roles from the pattern manager local filesystem.

```
POST /v1/upload/chef

Header: Authorization = Bearer $ACCESS_TOKEN

{
  "cookbooks": "True",
  "roles": "True",
  "source_repos": "file:///var/local_repo"
}
```

Responses

The following table lists responses that are returned for a request to upload cookbooks and roles to the pattern manager:

HTTP Code	Description	Schema
200	OK	No content
400	Bad Request	
403	The requester has not specified the correct access token in the Authorization HTTP header	
500	Internal server error	

List cookbooks and roles

List the cookbooks and roles that have been uploaded to the pattern manager

```
GET /v1/info/chef
```

Description

Return a list of all cookbooks and roles that have been uploaded to the pattern manager

Responses

The following table lists API responses:

HTTP Code	Description	Schema
200	Returns the list of cookbooks and roles that reside on the pattern manager.	See example

HTTP Code	Description	Schema
403	The requester has not specified the correct access token in the Authorization HTTP header.	
500	An internal error occurred while processing the request.	

Example

The following block is an example of a successful request:

```
Header: Authorization = Bearer $ACCESS_TOKEN

{
  "cookbooks": [
    "db2                0.1.36",
    "httpd              1.1.27",
    "ibm_cloud_utils    0.1.29"
  ],
  "recipes": [
    "db2",
    "db2::cleanup",
    "db2::create_db",
    "db2::fixpack",
    "db2::gather_evidence",
    "db2::harden",
    "db2::install",
    "db2::license",
    "db2::prereq",
    "db2::prereq_check",
    "httpd",
    "httpd::cleanup",
    "httpd::config_httpd_conf",
    "httpd::config_ssl",
    "httpd::config_vhosts",
    "httpd::gather_evidence",
    "httpd::install",
    "httpd::prereq",
    "httpd::service",
    "ibm_cloud_utils",
    "ibm_cloud_utils::gather_evidence",
    "ibm_cloud_utils::install",
    "ibm_cloud_utils::prereq"
  ],
  "roles": [
    "db2_create_db",
    "db2_v105_install",
    "db2_v111_install",
    "httpd24-base-install",
    "httpd24-ssl-vhosts"
  ]
}
```

Clear cookbooks and roles

Clears cookbooks and roles from the pattern manager.

```
POST /v1/clear_env/chef
```

Description

Deletes cookbooks and roles from the pattern manager. You can specify to delete cookbooks, roles, or both. By default all cookbooks and roles are deleted.

This API requires the pattern manager admin token.

Parameters

The following table lists the parameters that are used to clear cookbooks and roles from the pattern manager:

Type	Name	Description	Schema	Default	Required
Body	cookbooks	States whether cookbooks should be deleted	Boolean	True	No

Type	Name	Description	Schema	Default	Required
Body	pattern	Delete cookbooks and roles that match this pattern	String	•	No
Body	roles	States whether roles should be deleted	Boolean	True	No

Example Request

```
POST /v1/clear_env/chef

Header: Authorization = Bearer $ADMIN_TOKEN

{
  "cookbooks": "true",
  "roles": "true",
  "pattern": "db2.*"
}
```

Responses

The table lists responses that are returned for a request to clear cookbooks and roles from the pattern manager:

HTTP Code	Description	Schema
200	OK	No content
400	Bad Request	
403	The requester has not specified the correct access token in the Authorization HTTP header	
500	Internal server error	

Get service logs

Return the pattern manager service logs

```
GET /v1/admin/log
```

Description

Return the pattern manager service logs in one of three formats. The format returned is based on the value of the Accept HTTP header, with application/json being the default response format.

This request requires the pattern manager admin token. The following table lists the accept HTP header options:

Accept Header Value	Description
application/json	Filtered service log entries in JSON format
application/gzip	All service logs in a tar.gz file
application/zip	All service logs in a zip file
text/plain	The latest service log file

Parameters

These parameters are only valid when requesting a response in JSON format. If no parameters are specified, then the last 5000 entries in the service log file will be returned.

The following table lists the parameters that are used to filter service log entries returned in JSON response:

Table 2.

Type	Name	Description	Schema	Default	Required
Body	contains	Returns all log entries that contain the specified value	String		No
Body	end_dts	Date timestamp to end log entries collections. Accepted formats include YYYY-mm-dd HH:MM:SS,fff, YYYY-mm-dd HH:MM:SS, and YYYY-mm-dd	String		No
Body	levels	List of logging levels to include: CRITICAL, ERROR, WARNING, INFO, and DEBUG	Array		No

Type	Name	Description	Schema	Default	Required
Body	request_id	Returns entries associated with a specific request ID. Can be used to narrow logs down to a single request train	String		No
Body	remote	List of properties to remove from the JSON response including body, dts, level and request_id	Array		No
Body	start_dts	Date timestamp to start log entries collections. Accepted formats include YYYY-mm-dd HH:MM:SS,fff, YYYY-mm-dd HH:MM:SS, and YYYY-mm-dd	String		No

Example Request

```
Header: Authorization = Bearer $ADMIN_TOKEN

{
    "contains": "/v1/info/chef",
    "levels": ["DEBUG"],
    "start_dts": "2017-08-03 14:50:10"
}
```

Responses

The following table lists the API responses:

HTTP Code	Description	Schema
200	Returns the requested pattern manager service log entries	
403	The requester has not specified the correct admin token in the Authorization HTTP header.	
500	An internal error occurred while processing the request.	

Example Response

The following block is an example of a successful response in JSON format.

```
{
  log: [
    {
      body: "API response: Status = 200. Data log suppressed******* '/v1/info/chef GET ",
      dts: "2017-08-03 14:50:10,800",
      level: "DEBUG",
      request_id: "73fdad0f5f4d43829da05ad245e168da"
    },
    {
      body: "******* '/v1/info/chef GETExecuting cmd --> cd /tmp/tmpgkDyxz; knife ssl fetch ",
      dts: "2017-08-03 14:50:19,286",
      level: "DEBUG",
      request_id: "f7609ed9232548698c1f7c3c82dac9e7"
    }
  ]
}
```

Reference

View reference information about IBM Cloud Pak® for AIOps.

- Infrastructure Automation

Operator cluster service versions (CSVs)

For more information about the operator cluster service versions (CSVs) which are installed with IBM Cloud Pak for AIOps, see the following information.

You can use the following command to get the list of CSVs that are present in a Red Hat OpenShift Container Platform cluster.

```
oc get csv
```

The following table lists the CSVs, which are installed with IBM Cloud Pak for AIOps and present in a Red Hat OpenShift Container Platform cluster.

Important: Versions are correct at time of release, but newer patch versions may have been released subsequently.

Table. Operators

Name	Display	Version
aimanager-operator.v4.2.1	IBM AIOps AI Manager	`4.2.1`
aiopsedge-operator.v4.2.1	IBM AIOps Edge	`4.2.1`
asm-operator.v4.2.1	IBM Netcool Agile Service Manager	`4.2.1`
ibm-aiops-ir-ai.v4.2.1	IBM Watson AIOps Issue Resolution AI & Analytics	`4.2.1`
ibm-aiops-ir-core.v4.2.1	IBM Watson AIOps Issue Resolution Core	`4.2.1`
ibm-aiops-ir-lifecycle.v4.2.1	IBM Cloud Pak for Watson AIOps Lifecycle	`4.2.1`
ibm-aiops-orchestrator.v4.2.1	IBM Cloud Pak for AIOps	`4.2.1`
ibm-automation-elastic.v1.3.16	IBM Elastic	`1.3.16`
ibm-automation-flink.v1.3.16	IBM Automation Foundation Flink	`1.3.16`
ibm-common-service-operator.v3.23.12	IBM Cloud Pak foundational services	`3.23.12`
ibm-management-kong.v4.2.1	IBM Internal - IBM AIOps Kong	`4.2.1`
ibm-cloud-databases-redis.v1.6.11	IBM Operator for Redis	`1.6.11`
ibm-watson-aiops-ui-operator.v4.2.1	IBM AIOps UI	`4.2.1`
ibm-secure-tunnel-operator.v4.2.1	IBM Secure Tunnel	`4.2.1`
portworx-operator.v23.10.3	Portworx Enterprise	`23.10.3`

You can use the following command to get the subscriptions in a Red Hat OpenShift Container Platform cluster.

```
oc get sub
```

Permissions (IBM Cloud Pak for AIOps)

The following content shows the permissions and cluster permissions that are required by the IBM Cloud Pak for AIOps operators.

```
aimanager-operator:
  clusterPermissions:
  - rules:
    - apiGroups:
      - scheduling.k8s.io
      resources:
      - priorityclasses
      verbs:
      - create
      - delete
      - list
      - watch
      - get
      - patch
      - update
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - clusterroles
      - clusterrolebindings
      verbs:
      - get
      - watch
      - delete
      - list
    serviceAccountName: aimanager-operator-controller-manager
  permissions:
  - rules:
    - apiGroups:
      - ''
      resources:
      - configmaps
      verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
```

```
    - delete
  - apiGroups:
    - coordination.k8s.io
    resources:
    - leases
    verbs:
    - get
    - list
    - watch
    - create
    - update
    - patch
    - delete
  - apiGroups:
    - ''
    resources:
    - events
    verbs:
    - create
    - patch
  - apiGroups:
    - ''
    - policy
    - template.openshift.io
    resources:
    - pods
    - pods/exec
    - pods/log
    - services
    - services/finalizers
    - endpoints
    - persistentvolumeclaims
    - events
    - configmaps
    - secrets
    - serviceaccounts
    - serviceaccount
    - poddisruptionbudgets
    - processedtemplates
    verbs:
    - '*'
  - apiGroups:
    - coordination.k8s.io
    resources:
    - leases
    verbs:
    - get
    - list
    - watch
    - create
    - update
    - patch
    - delete
  - apiGroups:
    - monitoring.coreos.com
    resources:
    - servicemonitors
    verbs:
    - get
    - create
  - apiGroups:
    - rbac.authorization.k8s.io
    resources:
    - roles
    - rolebindings
    - clusterroles
    - clusterrolebindings
    verbs:
    - get
    - list
    - patch
    - update
    - delete
    - create
    - watch
  - apiGroups:
    - batch
    resources:
    - jobs
    - cronjobs
    verbs:
```

```
      - '*'
    - apiGroups:
      - apps
      resources:
      - deployments
      - deployments/finalizers
      - daemonsets
      - replicasets
      - statefulsets
      verbs:
      - '*'
    - apiGroups:
      - security.openshift.io
      resourceNames:
      - restricted
      resources:
      - securitycontextconstraints
      verbs:
      - use
    - apiGroups:
      - security.openshift.io
      resources:
      - securitycontextconstraints
      verbs:
      - create
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - monitoring.coreos.com
      resources:
      - servicemonitors
      verbs:
      - get
      - create
    - apiGroups:
      - apps
      resourceNames:
      - aimanager-operator
      resources:
      - deployments/finalizers
      verbs:
      - update
    - apiGroups:
      - ''
      resources:
      - pods
      verbs:
      - get
    - apiGroups:
      - apps
      resources:
      - replicasets
      - deployments
      verbs:
      - get
    - apiGroups:
      - cert-manager.io
      resources:
      - issuers
      - certificates
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - eventprocessing.automation.ibm.com
      resources:
      - eventprocessors
      verbs:
      - create
      - delete
      - get
      - list
      - patch
```

```
    - update
    - watch
  - apiGroups:
    - ibmevents.ibm.com
    resources:
    - kafkas
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - connectors.aiops.ibm.com
    resources:
    - '*'
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ai-manager.watson-aiops.ibm.com
    resources:
    - '*'
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - helm.operator-sdk
    resources:
    - IbmMinio
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - apiextensions.k8s.io
    resources:
    - '*'
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    - batch
    - extensions
    - apps
    - policy
    - rbac.authorization.k8s.io
    - autoscaling
    - route.openshift.io
    - authorization.openshift.io
    - networking.k8s.io
    - metrics.k8s.io
    resources:
    - pods
    - pods/log
    - poddisruptionbudgets
    - secrets
    - jobs
    - configmaps
    - deployments
    - daemonsets
    - statefulsets
    - replicasets
```

```
      - services
      - services/finalizers
      - persistentvolumeclaims
      - cronjobs
      - pods/exec
      - pods/portforward
      - serviceaccounts
      - namespaces
      - roles
      - rolebindings
      - horizontalpodautoscalers
      - routes
      - routes/custom-host
      - ingresses
      - endpoints
      - cronjob
      - networkpolicies
      - events
      - jobs/status
      - pods/status
      - resourcequotas
      - resourcequotas/status
      - nodes
      verbs:
      - apply
      - create
      - get
      - delete
      - watch
      - update
      - edit
      - exec
      - list
      - patch
      - deletecollection
    - apiGroups:
      - cpd.ibm.com
      resources:
      - cpdinstalls
      - cpdinstalls/spec
      - cpdinstalls/status
      verbs:
      - apply
      - create
      - delete
      - edit
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - build.openshift.io
      resources:
      - buildconfigs
      - buildconfigs/instantiate
      - buildconfigs/instantiatebinary
      - buildconfigs/webhooks
      - buildlogs
      - builds
      - builds/clone
      - builds/details
      - builds/log
      verbs:
      - create
      - delete
      - list
      - watch
      - get
      - patch
      - update
    - apiGroups:
      - image.openshift.io
      resources:
      - imagestreams
      - imagestreams/layers
      - imagestreams/secrets
      - imagestreams/status
      - imagestreamimages
      - imagestreamimports
      - imagestreammappings
```

```
      - imagestreamtags
      verbs:
      - create
      - delete
      - list
      - watch
      - get
      - patch
      - update
    - apiGroups:
      - apps
      resourceNames:
      - cpd-meta-operator
      resources:
      - deployments/finalizers
      verbs:
      - update
    - apiGroups:
      - ''
      resources:
      - pods
      verbs:
      - get
    - apiGroups:
      - apps
      resources:
      - replicasets
      - deployments
      verbs:
      - get
    - apiGroups:
      - metaoperator.cpd.ibm.com
      resources:
      - '*'
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - certmanager.k8s.io
      resources:
      - issuers
      - issuers/status
      - issuers/finalizers
      - certificates
      - certificates/status
      - certificates/finalizers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - operator.ibm.com
      resources:
      - operandrequests
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - zen.cpd.ibm.com
      resources:
      - '*'
      verbs:
      - create
      - delete
      - get
      - list
      - patch
```

```
      - update
      - watch
    - apiGroups:
      - admissionregistration.k8s.io
      resources:
      - validatingwebhookconfigurations
      - mutatingwebhookconfigurations
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    serviceAccountName: aimanager-operator-controller-manager
aiopsedge-operator:
  clusterPermissions:
  - rules:
    - apiGroups:
      - ''
      resourceNames:
      - console-config
      resources:
      - configmaps
      verbs:
      - get
      - watch
    - apiGroups:
      - apiextensions.k8s.io
      resources:
      - customresourcedefinitions
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - config.openshift.io
      resources:
      - clusterversions
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - config.openshift.io
      resources:
      - ingresses
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - aiopskafkatopics
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - aiopskafkatopics/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - bundlemanifests
      - bundlemanifests/finalizers
      verbs:
      - create
      - delete
      - get
```

```
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - bundlemanifests/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - operator.openshift.io
      resources:
      - dnses
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - storage.k8s.io
      resources:
      - storageclasses
      verbs:
      - get
      - list
      - watch
    serviceAccountName: manager
  permissions:
  - rules:
    - apiGroups:
      - ''
      resources:
      - configmaps
      verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
    - apiGroups:
      - ''
      resources:
      - configmaps/status
      verbs:
      - get
      - update
      - patch
    - apiGroups:
      - ''
      resources:
      - events
      verbs:
      - create
    - apiGroups:
      - ''
      resources:
      - '*'
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - ''
      resources:
      - configmaps
      - endpoints
      - secrets
      - serviceaccounts
      - services
      verbs:
      - create
      - delete
      - get
```

```
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - configmaps/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - configmaps/finalizers
    - secrets/finalizers
    - serviceaccounts/finalizers
    - services/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - events
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - pods
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - ''
    resources:
    - secrets
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - ''
    resources:
    - services
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - ''
    resources:
    - services/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - apidiscovery.ibm.com
```

```
    resources:
    - apis
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - apps
    resources:
    - '*'
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - apps
    resources:
    - deployments
    - replicasets
    - statefulsets
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - apps
    resources:
    - deployments/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - asm.ibm.com
    resources:
    - asmformations
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - autoscaling
    resources:
    - horizontalpodautoscalers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - build.openshift.io
    resources:
    - buildconfigs
    verbs:
    - delete
    - get
    - list
    - watch
  - apiGroups:
    - camel.apache.org
    resources:
    - camelcatalogs
```

```
    - integrationkits
    - integrationplatforms
    - integrations
    verbs:
    - delete
    - get
    - list
    - watch
  - apiGroups:
    - camel.apache.org
    resources:
    - integrations
    - kameletbindings
    - kamelets
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - cert-manager.io
    resources:
    - certificates
    verbs:
    - create
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - cert-manager.io
    resources:
    - certificates
    - issuers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - cert-manager.io
    resources:
    - issuers
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - connectors.aiops.ibm.com
    resources:
    - aiopsedges
    - aiopsedges/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - connectors.aiops.ibm.com
    resources:
    - aiopsedges/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - connectors.aiops.ibm.com
    resources:
    - connectorcomponents
    - connectorcomponents/finalizers
    verbs:
```

```
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - connectorcomponents
      - connectorconfigurations
      - connectorconfigurations/finalizers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - connectorcomponents/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - connectorconfigurations/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - connectorschemas
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - gitapps
      - gitapps/finalizers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - gitapps/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - connectors.aiops.ibm.com
      resources:
      - microedgeconfigurations
      - microedgeconfigurations/finalizers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
```

```
    - update
    - watch
  - apiGroups:
    - coordination.k8s.io
    resources:
    - leases
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ibmevents.ibm.com
    resources:
    - kafkas
    - kafkatopics
    - kafkausers
    - mirrormakers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - image.openshift.io
    resources:
    - imagestreams
    verbs:
    - delete
    - get
    - list
    - watch
  - apiGroups:
    - monitoring.coreos.com
    resources:
    - podmonitors
    - servicemonitors
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - network.openshift.io
    resources:
    - egressnetworkpolicies
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - networking.k8s.io
    resources:
    - networkpolicies
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - openliberty.io
    resources:
    - openlibertyapplications
    verbs:
    - create
```

```
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - operator.ibm.com
      resources:
      - operandconfigs
      - operandregistries
      - operandrequests
      verbs:
      - create
      - delete
      - deletecollection
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - operators.coreos.com
      resources:
      - operatorgroups
      - subscriptions
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - rolebindings
      - roles
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - route.openshift.io
      resources:
      - routes
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - route.openshift.io
      resources:
      - routes/custom-host
      verbs:
      - create
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - vault.aiops.ibm.com
      resources:
      - vaultaccesses
      - vaultdeploys
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    serviceAccountName: manager
asm-operator:
```

```
  clusterPermissions:
  - rules:
    - apiGroups:
      - scheduling.k8s.io
      resources:
      - priorityclasses
      verbs:
      - get
      - list
      - patch
      - update
      - delete
      - create
      - watch
    serviceAccountName: asm-operator
  permissions:
  - rules:
    - apiGroups:
      - policy
      resources:
      - poddisruptionbudgets
      verbs:
      - get
      - list
      - patch
      - update
      - delete
      - create
      - watch
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - roles
      - rolebindings
      - clusterroles
      - clusterrolebindings
      verbs:
      - get
      - list
      - patch
      - update
      - delete
      - create
      - watch
    - apiGroups:
      - autoscaling
      resources:
      - horizontalpodautoscalers
      verbs:
      - get
      - list
      - patch
      - update
      - delete
      - create
      - watch
    - apiGroups:
      - ''
      resources:
      - pods
      - services
      - services/finalizers
      - endpoints
      - persistentvolumeclaims
      - events
      - configmaps
      - secrets
      - role
      - rolebinding
      verbs:
      - get
      - list
      - patch
      - update
      - delete
      - create
      - watch
    - apiGroups:
      - ''
      - extensions
      resources:
```

```
      - deployments
      - configmaps
      - ingresses
      - services
      - serviceaccounts
      - persistentvolumeclaims
      verbs:
      - '*'
    - apiGroups:
      - ''
      resources:
      - namespaces
      verbs:
      - get
    - apiGroups:
      - apps
      resources:
      - deployments
      - deployments/finalizers
      - daemonsets
      - replicasets
      - statefulsets
      verbs:
      - '*'
    - apiGroups:
      - batch
      resources:
      - jobs
      - cronjobs
      verbs:
      - '*'
    - apiGroups:
      - route.openshift.io
      resources:
      - routes
      - routes/custom-host
      verbs:
      - '*'
    - apiGroups:
      - asm.ibm.com
      resources:
      - asms
      - asms/finalizers
      - Asms
      - Asm/status
      - Asms
      verbs:
      - '*'
    - apiGroups:
      - asm.ibm.com
      resources:
      - '*'
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - monitoring.coreos.com
      resources:
      - servicemonitors
      verbs:
      - get
      - create
    - apiGroups:
      - networking.k8s.io
      resources:
      - networkpolicies
      - ingresses
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
```

```
      - ''
      resources:
      - configmaps/status
      - endpoints
      verbs:
      - get
      - update
      - patch
    - apiGroups:
      - asm.ibm.com
      resources:
      - appdiscoes
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - asm.ibm.com
      resources:
      - appdiscoes/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - base.automation.ibm.com
      resources:
      - cartridgerequirements
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - zen.cpd.ibm.com
      resources:
      - zenservices
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - core.ir.aiops.ibm.com
      resources:
      - issueresolutioncores
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - cert-manager.io
      resources:
      - issuers
      - certificates
      verbs:
      - get
      - list
      - watch
      - create
      - patch
      - update
      - delete
    serviceAccountName: asm-operator
ibm-aiops-ir-ai:
  permissions:
  - rules:
    - apiGroups:
      - coordination.k8s.io
      resources:
      - leases
      verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
```

```
  - apiGroups:
    - ''
    resources:
    - configmaps
    verbs:
    - get
    - list
    - watch
    - create
    - update
    - patch
    - delete
  - apiGroups:
    - ''
    resources:
    - configmaps/status
    verbs:
    - get
    - update
    - patch
  - apiGroups:
    - ''
    resources:
    - events
    verbs:
    - create
    - patch
  - apiGroups:
    - ai-manager.watson-aiops.ibm.com
    resources:
    - algorithms
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ai.ir.aiops.ibm.com
    resources:
    - aiopsanalyticsorchestrators
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ai.ir.aiops.ibm.com
    resources:
    - aiopsanalyticsorchestrators/finalizers
    verbs:
    - update
  - apiGroups:
    - ai.ir.aiops.ibm.com
    resources:
    - aiopsanalyticsorchestrators/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - apps
    resources:
    - deployments
    verbs:
    - create
    - delete
    - get
    - list
    - update
    - watch
  - apiGroups:
    - apps
    resources:
    - statefulsets
    verbs:
```

```
      - create
      - delete
      - get
      - list
      - update
      - watch
    - apiGroups:
      - batch
      resources:
      - jobs
      verbs:
      - create
      - delete
      - get
      - list
      - update
      - watch
    - apiGroups:
      - ''
      resources:
      - configmaps
      verbs:
      - create
      - delete
      - get
      - update
      - watch
    - apiGroups:
      - ''
      resources:
      - persistentvolumeclaims
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - ''
      resources:
      - secrets
      verbs:
      - create
      - delete
      - get
      - list
      - update
      - watch
    - apiGroups:
      - ''
      resources:
      - serviceaccounts
      verbs:
      - create
      - delete
      - get
      - update
    - apiGroups:
      - ''
      resources:
      - services
      verbs:
      - create
      - delete
      - get
      - update
    - apiGroups:
      - networking.k8s.io
      resources:
      - networkpolicies
      verbs:
      - create
      - delete
      - get
      - update
    serviceAccountName: ibm-ir-ai-operator-controller-management-sa
ibm-aiops-ir-core:
  permissions:
  - rules:
```

```
  - apiGroups:
    - ''
    - coordination.k8s.io
    resources:
    - configmaps
    - leases
    verbs:
    - get
    - list
    - watch
    - create
    - update
    - patch
    - delete
  - apiGroups:
    - ''
    resources:
    - events
    verbs:
    - create
    - patch
  - apiGroups:
    - apps
    resources:
    - deployments
    verbs:
    - create
    - delete
    - get
    - list
    - update
    - watch
  - apiGroups:
    - apps
    resources:
    - statefulsets
    verbs:
    - create
    - delete
    - get
    - list
    - update
    - watch
  - apiGroups:
    - batch
    resources:
    - cronjobs
    verbs:
    - create
    - delete
    - get
    - list
    - update
    - watch
  - apiGroups:
    - batch
    resources:
    - jobs
    verbs:
    - create
    - delete
    - get
    - list
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - configmaps
    verbs:
    - create
    - delete
    - get
    - list
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - persistentvolumeclaims
    verbs:
```

```
      - create
      - delete
      - get
      - list
      - update
      - watch
    - apiGroups:
      - ''
      resources:
      - pods
      verbs:
      - list
      - watch
    - apiGroups:
      - ''
      resources:
      - secrets
      verbs:
      - create
      - delete
      - get
      - list
      - update
      - watch
    - apiGroups:
      - ''
      resources:
      - serviceaccounts
      verbs:
      - create
      - delete
      - get
      - list
      - update
    - apiGroups:
      - ''
      resources:
      - services
      verbs:
      - create
      - delete
      - get
      - list
      - update
    - apiGroups:
      - core.ir.aiops.ibm.com
      resources:
      - issueresolutioncores
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - core.ir.aiops.ibm.com
      resources:
      - issueresolutioncores/finalizers
      verbs:
      - update
    - apiGroups:
      - core.ir.aiops.ibm.com
      resources:
      - issueresolutioncores/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - networking.k8s.io
      resources:
      - networkpolicies
      verbs:
      - create
      - delete
      - get
      - update
    serviceAccountName: ir-core-operator-controller-manager
ibm-aiops-ir-lifecycle:
```

```
  permissions:
  - rules:
    - apiGroups:
      - ''
      resources:
      - configmaps
      verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
    - apiGroups:
      - coordination.k8s.io
      resources:
      - leases
      verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
    - apiGroups:
      - ''
      resources:
      - events
      verbs:
      - create
      - patch
    - apiGroups:
      - apps
      resources:
      - deployments
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - apps
      resources:
      - statefulsets
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - base.automation.ibm.com
      resources:
      - cartridgerequirements
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - batch
      resources:
      - cronjobs
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - batch
      resources:
      - jobs
```

```
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - cert-manager.io
    resources:
    - certificates
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - cert-manager.io
    resources:
    - issuers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - certmanager.k8s.io
    resources:
    - certificates
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - configmaps
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - endpoints
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - ''
    resources:
    - persistentvolumeclaims
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - pods
    verbs:
```

```
    - get
    - list
    - watch
  - apiGroups:
    - ''
    resources:
    - secrets
    - serviceaccounts
    - services
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - eventprocessing.automation.ibm.com
    resources:
    - eventprocessors
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - flink.automation.ibm.com
    resources:
    - flinkclusters
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ibmevents.ibm.com
    resources:
    - kafkausers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - lifecycle.ir.aiops.ibm.com
    resources:
    - lifecycleservices
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - lifecycle.ir.aiops.ibm.com
    resources:
    - lifecycleservices/finalizers
    verbs:
    - update
  - apiGroups:
    - lifecycle.ir.aiops.ibm.com
    resources:
    - lifecycleservices/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - lifecycle.ir.aiops.ibm.com
```

```
      resources:
      - lifecycletriggers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - lifecycle.ir.aiops.ibm.com
      resources:
      - lifecycletriggers/finalizers
      verbs:
      - update
    - apiGroups:
      - lifecycle.ir.aiops.ibm.com
      resources:
      - lifecycletriggers/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - monitoring.coreos.com
      resources:
      - servicemonitors
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - networking.k8s.io
      resources:
      - networkpolicies
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - route.openshift.io
      resources:
      - routes
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    serviceAccountName: ir-lifecycle-operator-controller-manager
ibm-aiops-orchestrator:
  permissions:
  - rules:
    - apiGroups:
      - ''
      resources:
      - configmaps
      verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
    - apiGroups:
      - coordination.k8s.io
      resources:
      - leases
      verbs:
```

```
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
    - apiGroups:
      - ''
      resources:
      - events
      verbs:
      - create
      - patch
    - apiGroups:
      - ai-manager.watson-aiops.ibm.com
      resources:
      - aimanagers
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - ai.ir.aiops.ibm.com
      resources:
      - aiopsanalyticsorchestrators
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - asm.ibm.com
      resources:
      - asms
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - base.automation.ibm.com
      resources:
      - automationbases
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - base.automation.ibm.com
      resources:
      - cartridgerequirements
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - base.automation.ibm.com
      resources:
      - cartridgesrequirements/finalizers
      verbs:
      - update
    - apiGroups:
      - cert-manager.io
      resources:
      - certificates
      - issuers
      verbs:
      - create
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
```

```
    - connectors.aiops.ibm.com
    resources:
    - aiopsedges
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - aiopsuis
    - baseuis
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - ''
    resources:
    - configmaps
    - persistentvolumeclaims
    - secrets
    - serviceaccounts
    - services
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - pods
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - core.automation.ibm.com
    resources:
    - automationuiconfigs
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - core.automation.ibm.com
    resources:
    - cartridges
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - core.automation.ibm.com
    resources:
    - cartridges/finalizers
    verbs:
    - update
  - apiGroups:
    - core.ir.aiops.ibm.com
    resources:
    - issueresolutioncores
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - elastic.automation.ibm.com
    resources:
```

```
    - elasticsearches
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - eventprocessing.automation.ibm.com
    resources:
    - eventprocessors
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - flink.automation.ibm.com
    resources:
    - flinkclusters
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ibmevents.ibm.com
    resources:
    - kafkas
    - kafkausers
    verbs:
    - create
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - lifecycle.ir.aiops.ibm.com
    resources:
    - lifecycleservices
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - management.ibm.com
    resources:
    - kongs
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - modeltrain.ibm.com
    resources:
    - modeltrains
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - networking.k8s.io
    resources:
    - networkpolicies
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
```

```
  - apiGroups:
    - operator.ibm.com
    resources:
    - commonservices
    - operandbindinfos
    - operandconfigs
    - operandregistries
    - operandrequests
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - operators.coreos.com
    resources:
    - clusterserviceversions
    verbs:
    - delete
    - get
    - list
    - watch
  - apiGroups:
    - operators.coreos.com
    resources:
    - subscriptions
    verbs:
    - delete
    - get
    - list
    - watch
  - apiGroups:
    - orchestrator.aiops.ibm.com
    resources:
    - customsizes
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - orchestrator.aiops.ibm.com
    resources:
    - customsizes/finalizers
    verbs:
    - update
  - apiGroups:
    - orchestrator.aiops.ibm.com
    resources:
    - customsizes/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - orchestrator.aiops.ibm.com
    resources:
    - installations
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - orchestrator.aiops.ibm.com
    resources:
    - installations/finalizers
    verbs:
    - update
  - apiGroups:
    - orchestrator.aiops.ibm.com
    resources:
```

```
      - installations/status
      verbs:
      - get
      - patch
      - update
    - apiGroups:
      - postgres.aiops.ibm.com
      resources:
      - postgresdbs
      - postgreservices
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - postgresql.k8s.enterprisedb.io
      resources:
      - clusters
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - rolebindings
      - roles
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - redis.databases.cloud.ibm.com
      resources:
      - formations
      - redissentinels
      verbs:
      - delete
      - get
      - list
      - watch
    - apiGroups:
      - shim.bedrock.ibm.com
      resources:
      - kafkaclaims
      verbs:
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - shim.bedrock.ibm.com
      resources:
      - kafkaclaims/finalizers
      verbs:
      - update
    - apiGroups:
      - sretooling.management.ibm.com
      resources:
      - tunnels
      verbs:
      - get
      - list
      - watch
    - apiGroups:
      - zen.cpd.ibm.com
      resources:
      - zenservices
      verbs:
      - get
      - list
      - watch
    serviceAccountName: ibm-aiops-orchestrator-controller-manager
ibm-management-kong:
  clusterPermissions:
```

```
  - rules:
    - apiGroups:
      - extensions
      resources:
      - ingresses
      verbs:
      - '*'
    - apiGroups:
      - networking.k8s.io
      resources:
      - ingresses
      - ingressclasses
      - networkpolicies
      verbs:
      - '*'
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - clusterroles
      - roles
      - clusterrolebindings
      - rolebindings
      verbs:
      - '*'
    - apiGroups:
      - apiextensions.k8s.io
      resources:
      - customresourcedefinitions
      verbs:
      - '*'
    - apiGroups:
      - batch
      resources:
      - jobs
      - cronjobs
      verbs:
      - '*'
    - apiGroups:
      - ''
      resources:
      - pods
      - services
      - services/finalizers
      - endpoints
      - persistentvolumeclaims
      - events
      - configmaps
      - secrets
      - serviceaccounts
      verbs:
      - '*'
    - apiGroups:
      - apps
      resources:
      - deployments
      - daemonsets
      - replicasets
      - statefulsets
      verbs:
      - '*'
    - apiGroups:
      - ''
      resources:
      - namespaces
      verbs:
      - get
    - apiGroups:
      - ''
      resources:
      - configmaps
      - secrets
      verbs:
      - '*'
    - apiGroups:
      - charts.konghq.com
      resources:
      - '*'
      verbs:
      - '*'
    - apiGroups:
      - ''
```

```
      resources:
      - events
      verbs:
      - create
    - apiGroups:
      - monitoring.coreos.com
      resources:
      - servicemonitors
      verbs:
      - get
      - create
    - apiGroups:
      - apps
      resourceNames:
      - ibm-kong-operator
      resources:
      - deployments/finalizers
      verbs:
      - update
    - apiGroups:
      - ''
      resources:
      - pods
      verbs:
      - get
    - apiGroups:
      - apps
      resources:
      - replicasets
      - deployments
      verbs:
      - get
    serviceAccountName: ibm-kong-operator
  permissions:
  - rules:
    - apiGroups:
      - management.ibm.com
      resources:
      - '*'
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - zen.cpd.ibm.com
      resources:
      - zenextensions
      verbs:
      - '*'
    - apiGroups:
      - zen.cpd.ibm.com
      resources:
      - zenservices
      verbs:
      - get
      - list
    serviceAccountName: ibm-kong-operator
ibm-secure-tunnel-operator:
  clusterPermissions:
  - rules:
    - apiGroups:
      - ''
      resources:
      - pods
      verbs:
      - get
      - list
      - watch
      - update
    - apiGroups:
      - ''
      resources:
      - configmaps
      - endpoints
      - events
      - secrets
      - serviceaccounts
```

```
      - services
      - services/finalizers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - apiextensions.k8s.io
      resources:
      - customresourcedefinitions
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - apps
      resources:
      - deployments
      - replicasets
      - statefulsets
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - authentication.k8s.io
      resources:
      - tokenreviews
      verbs:
      - create
      - patch
    - apiGroups:
      - authorization.k8s.io
      resources:
      - subjectaccessreviews
      verbs:
      - create
      - patch
    - apiGroups:
      - extensions
      resources:
      - ingresses
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - monitoring.coreos.com
      resources:
      - servicemonitors
      verbs:
      - create
      - get
    - apiGroups:
      - networking.k8s.io
      resources:
      - ingresses
      - networkpolicies
      - ingressclasses
      verbs:
      - create
      - delete
      - get
      - list
      - patch
```

```
      - update
      - watch
    - apiGroups:
      - oauth.openshift.io
      resources:
      - oauthclients
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - operator.openshift.io
      resources:
      - dnses
      verbs:
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - clusterrolebindings
      - clusterroles
      - rolebindings
      - roles
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - route.openshift.io
      resources:
      - routes
      - routes/custom-host
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - securetunnel.management.ibm.com
      resources:
      - applicationmappings
      - applicationmappings/finalizers
      - applicationmappings/status
      - applications/finalizers
      - templates
      - templates/finalizers
      - templates/status
      - tunnelconnections
      - tunnelconnections/finalizers
      - tunnelconnections/status
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - sretooling.management.ibm.com
      resources:
      - tunnels
      - tunnels/finalizers
      - tunnels/status
      verbs:
      - create
```

```
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - tunnel.management.ibm.com
      resources:
      - applications
      - applications/finalizers
      - applications/status
      - templates
      - templates/finalizers
      - templates/status
      - tunnelconnections
      - tunnelconnections/finalizers
      - tunnelconnections/status
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - cert-manager.io
      resources:
      - certificates
      - issuers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - zen.cpd.ibm.com
      resources:
      - zenextensions
      - zenservices
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    serviceAccountName: ibm-secure-tunnel-operator
ibm-watson-aiops-ui-operator:
  permissions:
  - rules:
    - apiGroups:
      - ''
      resources:
      - configmaps
      verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
    - apiGroups:
      - ''
      resources:
      - configmaps/status
      verbs:
      - get
      - update
      - patch
    - apiGroups:
      - ''
      resources:
      - events
      verbs:
```

```
    - create
    - patch
  - apiGroups:
    - coordination.k8s.io
    resources:
    - leases
    verbs:
    - get
    - create
    - update
  - apiGroups:
    - ''
    resources:
    - configmaps
    - pods
    verbs:
    - create
    - delete
    - deletecollection
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - configmaps
    - secrets
    - services
    - services/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - configmaps
    - serviceaccounts
    - services
    - services/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - configmaps
    - services
    - services/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - apps
    resources:
    - deployments
    - deployments/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
```

```
  - apiGroups:
    - asm.ibm.com
    resources:
    - asms
    verbs:
    - get
    - list
  - apiGroups:
    - batch
    resources:
    - jobs
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - connectors.aiops.ibm.com
    resources:
    - connectorschemas
    verbs:
    - get
    - list
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - aimodeluis
    - aimodeluis/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - aimodeluis/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - aiopsuiextensions
    - aiopsuiextensions/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - aiopsuiextensions/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - aiopsuis
    - aiopsuis/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
```

```
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - aiopsuis/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - baseuis
    - baseuis/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - baseuis/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - connectoruis
    - connectoruis/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - connectoruis/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - insightsuis
    - insightsuis/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - consoleui.aiops.ibm.com
    resources:
    - insightsuis/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - ''
    resources:
    - configmaps
    verbs:
    - create
    - delete
    - get
    - update
  - apiGroups:
```

```
      - networking.k8s.io
      resources:
      - networkpolicies
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - operator.ibm.com
      resources:
      - operandrequests
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - rolebindings
      - roles
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - route.openshift.io
      resources:
      - routes
      verbs:
      - get
    - apiGroups:
      - zen.cpd.ibm.com
      resources:
      - zenextensions
      - zenextensions/status
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - zen.cpd.ibm.com
      resources:
      - zenservices
      verbs:
      - get
      - list
      - watch
    serviceAccountName: ibm-watson-aiops-ui-operator-manager-role
```

Permissions (Infrastructure Automation)

The following content shows the permissions and cluster permissions that are required by the Infrastructure Automation operators.

```
ibm-infrastructure-automation-operator:
  permissions:
  - rules:
    - apiGroups:
      - ''
      resources:
      - configmaps
      verbs:
```

```
    - get
    - list
    - watch
    - create
    - update
    - patch
    - delete
  - apiGroups:
    - coordination.k8s.io
    resources:
    - leases
    verbs:
    - get
    - list
    - watch
    - create
    - update
    - patch
    - delete
  - apiGroups:
    - ''
    resources:
    - events
    verbs:
    - create
    - patch
  - apiGroups:
    - ''
    resources:
    - serviceaccounts
    - services
    - services/finalizers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ''
    resources:
    - secrets
    verbs:
    - create
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - aiops.ibm.com
    resources:
    - iaconfigs
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - aiops.ibm.com
    resources:
    - iaconfigs/finalizers
    verbs:
    - update
  - apiGroups:
    - aiops.ibm.com
    resources:
    - iaconfigs/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - apps
    resources:
    - deployments
```

```
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - base.automation.ibm.com
    resources:
    - automationbases
    - cartridgerequirements
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - core.automation.ibm.com
    resources:
    - automationuiconfigs
    - cartridges
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - operator.ibm.com
    resources:
    - commonservices
    - operandbindinfos
    - operandconfigs
    - operandregistries
    - operandrequests
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - operators.coreos.com
    resources:
    - catalogsources
    - clusterserviceversions
    - subscriptions
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - zen.cpd.ibm.com
    resources:
    - zenservices
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - cert-manager.io
    resources:
    - issuers
    - certificates
```

```
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    serviceAccountName: ibm-infrastructure-automation-operator-controller-manager
ibm-management-cam-install:
  clusterPermissions:
  - rules:
    - apiGroups:
      - coordination.k8s.io
      resources:
      - leases
      verbs:
      - get
      - list
      - watch
      - create
      - update
      - patch
      - delete
    - apiGroups:
      - apiextensions.k8s.io
      resources:
      - customresourcedefinitions
      verbs:
      - '*'
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - clusterrolebindings
      - clusterroles
      verbs:
      - '*'
    - apiGroups:
      - certmanager.k8s.io
      resources:
      - clusterissuers
      verbs:
      - use
    - apiGroups:
      - certmanager.k8s.io
      resources:
      - issuers
      verbs:
      - use
      - get
      - create
      - delete
      - patch
      - list
    - apiGroups:
      - ibmcpcs.ibm.com
      resources:
      - secretshares
      verbs:
      - use
      - get
      - create
      - delete
      - patch
    - apiGroups:
      - policy
      resources:
      - podsecuritypolicies
      verbs:
      - '*'
    - apiGroups:
      - extensions
      resourceNames:
      - cam-services-psp
      resources:
      - podsecuritypolicies
      verbs:
      - '*'
    - apiGroups:
      - security.openshift.io
      resources:
```

```
    - securitycontextconstraints
    verbs:
    - get
    - delete
    - create
    - watch
    - list
  - apiGroups:
    - route.openshift.io
    resources:
    - routes
    - routes/custom-host
    verbs:
    - '*'
  - apiGroups:
    - foundation.ibm.com
    resources:
    - navconfigurations
    verbs:
    - '*'
  - apiGroups:
    - operator.ibm.com
    resources:
    - operandrequests
    verbs:
    - '*'
  serviceAccountName: cam-install-operator-controller-manager
permissions:
- rules:
  - apiGroups:
    - ''
    resources:
    - services
    - services/finalizers
    - endpoints
    - persistentvolumeclaims
    - events
    - configmaps
    - secrets
    - serviceaccounts
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - apps
    resources:
    - deployments
    - daemonsets
    - replicasets
    - statefulsets
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - monitoring.coreos.com
    resources:
    - servicemonitors
    verbs:
    - get
    - create
  - apiGroups:
    - apps
    resourceNames:
    - cam-install-operator-controller-manager
    resources:
    - deployments/finalizers
    verbs:
    - update
  - apiGroups:
    - ''
    resources:
```

```
      - pods
      verbs:
      - get
    - apiGroups:
      - apps
      resources:
      - replicasets
      - deployments
      verbs:
      - get
    - apiGroups:
      - networking.k8s.io
      resources:
      - networkpolicies
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - roles
      - rolebindings
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - cam.management.ibm.com
      resources:
      - manageservices
      - manageservices/finalizers
      - manageservices/status
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - certmanager.k8s.io
      resources:
      - certificates
      - issuers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - networking.k8s.io
      resources:
      - NetworkPolicy
      - ingresses
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - batch
      resources:
      - jobs
      verbs:
      - create
```

```
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - oidc.security.ibm.com
      resources:
      - clients
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    serviceAccountName: cam-install-operator-controller-manager
ibm-management-im-install:
  clusterPermissions:
  - rules:
    - apiGroups:
      - foundation.ibm.com
      resources:
      - multicluster-hub-nav
      - navconfigurations
      verbs:
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - infra.management.ibm.com
      resources:
      - connections
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - rbac.authorization.k8s.io
      resourceNames:
      - ibm-infra-management-install-operator-connection
      - ibm-infra-management-install-operator-navigation
      resources:
      - clusterrole
      verbs:
      - get
      - list
      - patch
      - update
    serviceAccountName: ibm-infra-management-install-operator
  permissions:
  - rules:
    - apiGroups:
      - ''
      resources:
      - configmaps
      - events
      - persistentvolumeclaims
      - pods
      - pods/finalizers
      - secrets
      - serviceaccounts
      - services
      - services/finalizers
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
```

```
    - ''
    resources:
    - pods/log
    verbs:
    - get
  - apiGroups:
    - apps
    resources:
    - deployments
    - deployments/scale
    - replicasets
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - apps
    resourceNames:
    - ibm-infra-management-application
    - ibm-infra-management-install-operator
    resources:
    - deployments/finalizers
    verbs:
    - update
  - apiGroups:
    - cert-manager.io
    resources:
    - certificates
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
  - apiGroups:
    - coordination.k8s.io
    resources:
    - leases
    verbs:
    - create
    - delete
    - get
    - list
    - update
  - apiGroups:
    - extensions
    resources:
    - deployments
    - deployments/scale
    - networkpolicies
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - ibmevents.ibm.com
    resources:
    - kafkas
    - kafkatopics
    - kafkausers
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - infra.management.ibm.com
    resources:
    - iminstalls
```

```
    verbs:
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - infra.management.ibm.com
    resources:
    - iminstalls/finalizers
    verbs:
    - update
  - apiGroups:
    - infra.management.ibm.com
    resources:
    - iminstalls/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - manageiq.org
    resources:
    - manageiqs
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - manageiq.org
    resources:
    - manageiqs/finalizers
    verbs:
    - update
  - apiGroups:
    - manageiq.org
    resources:
    - manageiqs/status
    verbs:
    - get
    - patch
    - update
  - apiGroups:
    - monitoring.coreos.com
    resources:
    - servicemonitors
    verbs:
    - create
    - get
  - apiGroups:
    - networking.k8s.io
    resources:
    - ingresses
    - networkpolicies
    verbs:
    - create
    - delete
    - get
    - list
    - patch
    - update
    - watch
  - apiGroups:
    - oidc.security.ibm.com
    resources:
    - clients
    verbs:
    - create
    - delete
    - get
    - patch
    - update
  - apiGroups:
    - operator.ibm.com
    resources:
    - operandrequests
    verbs:
```

```
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - rbac.authorization.k8s.io
      resources:
      - rolebindings
      - roles
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    - apiGroups:
      - route.openshift.io
      resources:
      - routes
      - routes/custom-host
      verbs:
      - create
      - delete
      - get
      - list
      - patch
      - update
      - watch
    serviceAccountName: ibm-infra-management-install-operator
```

Install commands requiring root or sudo (IBM Cloud Pak for AIOps)

The following content shows the commands that require root or sudo access for IBM Cloud Pak for AIOps

Online installs

Online installations of IBM Cloud Pak for AIOps can be run entirely as a non-root user, and do not require that user to have sudo access.

Offline installs

The following commands that are run during an air-gapped (offline) installation of IBM Cloud Pak for AIOps must be run as a root user, or by a user with sudo access.

```
yum install podman
```

```
podman login $LOCAL_DOCKER_REGISTRY
```

Alert and Event schemas

See the following tables to understand Cloud Pak for AIOps alert and event properties.

Cloud Pak for AIOps Alert schema

An alert is a mutable record of something that has happened over a certain period of time. It indicates an anomalous condition.

Cloud Pak for AIOps alert property	Description	Example
id	A system-generated unique id for the alert.	1a2a6787-59ad-4acd-bd0d-46c1ddfd8e06
state	The current state of this alert.	Open

Cloud Pak for AIOps alert property	Description	Example
eventCount	The count of events that have contributed to this alert.	2
acknowledged	Flag indicating alert acknowledgement.	false
team	Team with responsibility for the alert.	Db2 admin group
owner	Individual with responsibility for the alert.	user123
deduplicationKey	The key that uniquely identifies the anomalous condition that this alert represents.	{hostname=users-db.myapp.example.com, name=users-db.myapp.example.com, service=users-db.myapp.example.com, type=Service}-High query timeout count:++:Database query timeout-
signature	An id which uniquely identifies an anomalous condition on a resource. This may be used to identify historic alerts which refer to the same condition on the same resource.	{hostname=users-db.myapp.example.com, name=users-db.myapp.example.com, service=users-db.myapp.example.com, type=Service}-High query timeout count:++:Database query timeout-
firstOccurrenceTime	The time at which the anomalous condition was first detected, in millisecond granularity. Format: yyyy-mm-ddThh:mm:ss.sssZ (ISO8601)	2022-06-14 3:14:10.862
lastOccurrenceTime	The time at which the anomalous condition was last detected, in millisecond granularity. Format: yyyy-mm-ddThh:mm:ss.sssZ (ISO8601)	2022-06-14 3:14:10.862
lastStateChangeTime	The time in millisecond granularity for which the alert was last updated.	2022-06-14 3:14:10.862
summary	A human-readable description of the event.	Fan failure detected on server1.example.com.
langId	ISO 639-3 encoding if not specifed assumption is eng.	eng
severity	Indicates the alert severity level, which indicates how the perceived capability of the managed object has been affected.	5 - Major
sender	Identifies the resource which sent this alert. This may be the resource that is the subject of the event, or it may be some external system which is monitoring the subject.	{"service":"DemoMon","name": "hyper.example.com","type":"Netcool/OMNIbus"}
resource	Identifies the resource that is the subject of this alert. This is made up of multiple fields that describe the resource. There are a set of standard fields which should be used where applicable, but additional fields may be set.	{"service":"hyper.example.com","name": "hyper.example.com","hostname":"hyper.example.com", "type":"Service"}
expirySeconds	The number of seconds before the event should automatically expire. A value of zero indicates that there is no expiration.	0

Cloud Pak for AIOps alert property	Description	Example
links	An optional array of links to external systems which provide addition information or control over the event, or its subject. This may include a management console for the subject resource, or a page containing further event information from the source monitoring system. Each link should be in the form of a fully qualified URL.	[{"url":"`https://fan-controller.example.com/?fanId=1234`","linkType": "webpage","name": "management-console", "description":"Fan controller management console"}]
details	Additional properties that describe the alert.	{"additionalProp1": "string","additionalProp2":"string", "additionalProp3":"string"}
insights	An array of insights associated with the alert with one or more other entities or groupings.	insights.topology, insights.runbook, insights.scopeGroup
relatedIncidentIds	The set of incident id's that this alert is a member of either as a trigger or related context.	["related-incident-2","related-incident-1","related-incident-3"]

Cloud Pak for AIOps Event schema

An event is an immutable record of something happening at a certain time point. It may or may not indicate an anomalous condition.

Cloud Pak for AIOps event property	Description	Example
id	A system-generated unique id for the event.	1a2a6787-59ad-4acd-bd0d-46c1ddfd8e06
occurrenceTime	The time at which this event occurred.	6/15/2022, 3:14:10 PM
summary	A human-readable description of the event.	Fan failure detected on server1.example.com.
severity	Indicates the event severity level, which indicates how the perceived capability of the managed object has been affected.	5 - Major
sender	Identifies the resource which sent this event. This may be the resource that is the subject of the event, or it may be some external system which is monitoring the subject.	{"service":"DemoMon","name": "hyper.example.com","type":"Netcool/OMNIbus"}
resource	Identifies the resource that is the subject of this event. This is made up of multiple fields that describe the resource. There are a set of standard fields which should be used where applicable, but additional fields may be set.	{"service":"hyper.example.com","name": "hyper.example.com","hostname": "hyper.example.com", "type":"Service"} . For additional fields in property, see Resource property.
expirySeconds	The number of seconds before the event should automatically expire. A value of zero indicates that there is no expiration.	0

Cloud Pak for AIOps event property	Description	Example
links	An optional array of links to external systems which provide addition information or control over the event, or its subject. This may include a management console for the subject resource, or a page containing further event information from the source monitoring system. Each link should be in the form of a fully qualified URL.	[{"url":"**https://fan-controller.example.com/?fanId=1234**","linkType": "webpage","name":"management-console", "description":"Fan controller management console"}] . For additional fields in property, see Links property.
details	Additional properties that describe the event.	{"additionalProp1":"string","additionalProp2":"string", "additionalProp3":"string"}
insights	An array of insights associated with the event with one or more other entities or groupings.	insights.topology, insights.runbook, insights.scopeGroup. . For additional fields in property, see Insights property.
type	Values in an object that indicate the type of an event, such as a problem event or a resolution event, and the classification of an event.	{ "classification":":++:CEASelfMonitoring", "eventType":"problem"}. For additional fields in property, see EventType properties.

Alerts and Events schema fields

On this page, find fields that can be included in an Event or Alert schema in IBM Cloud Pak for AIOps for these properties:

- EventType
- Insights
- Links
- Resource

Note: format type used is Open API v3 definition.

EventType property schema

```
EventType:
      description: >
        The classification of the event, i.e. a description of what happened.
      type: object
      properties:
        eventType:
          description: >
            String description of whether the event is a problem
            or resolution event.
          type: string
          enum:
            - problem
            - resolution
          default: problem
        classification:
          description: >
            Description of the type of the event. E.g. Utilization, System
            status, Threshold breach
          type: string
          example: Utilization
        condition:
          description: >
            The condition/status or threshold causing the event. E.g. Down, 95%,
            Unavailable
          type: string
          example: "> 30 seconds"
```

Insights property schema

```
Insight:
      description: >
```

```
        Represents an insight reference to one or more other entities provided
        by a specified source that is related to associated element.
      type: object
      properties:
        id:
          description: >
            The unique identifier for this specific relationship, used to
            identify other related entities. Entities with the same insight
            id will participate in the same insight (i.e. this may act as a
            correlation key).
          type: string
          example: 98aa5f81-7da7-4063-bd66-dccaf59097d1
        type:
          description: The type of insight being described
          type: string
          example: ibm.com/causal-alert-relationship
        source:
          description: The source of the insight information
          type: string
          example: ibm.com/causal-correlation/temporal
        details:
          description: >
            Additional details about this insight. The schema of this object
            depends on the insight type.
          type: object
          $ref: "#/components/schemas/InsightDetails"

    AssociatedChange:
      type: object
      properties:
        link:
          type: string
          description: The URL to the associated change in source system.

    Conversation:
      type: object
      properties:
        link:
          type: string
          description: The URL to the conversation in source system.

    Grouping:
      type: object
      properties:
        policyId:
          type: string
          description: The id of the policy that creates the grouping.
        groupDetails:
          type: object
          description: >
            Details or links on group and/or to explain the reasons for the
            grouping.
    Runbook:
      type: object
      additionalProperties: false
      required:
        - policyId
        - runbookId
        - isAutomated
      properties:
        policyId:
          type: string
          description: The id of the policy that connected this insight.
        runbookId:
          type: string
          format: uuid
          description: The ID of the runbook
        runbookVersion:
          type: integer
          minimum: 1
          description: Optional. Defines whether a fixed version of a runbook should be linked. If, not
specified latest version is used.
        order:
          type: integer
          minimum: 1
          description: Optional. Order for runbook actions. Defines in which order runbooks should be
executed.
        isAutomated:
          type: boolean
          description: Defines whether or not the connected Runbook will be automatically executed, once the
policy matches.
```

```
        mappings:
          type: array
          description: In order to execute a runbook all required runbook parameters need to be filled out.
If the incident wants to define the values based on the policy or on its own attributes a suitable mapping
has to be created. This is required if automatically triggered execution is desired.
          items:
            type: object
            additionalProperties: false
            properties:
              mappingType:
                type: string
                description: Choose how a parameter is mapped. 'parameter' means using a field/property from
the Incident. 'fixed' means the same value will always be used. 'useDefault' will use the default from the
runbook. 'runtime' leave entry up to the operator.
                enum:
                  - parameter
                  - fixed
                  - useDefault
                  - runtime
              runbookParameterName:
                type: string
                description: The name of the runbook parameter as valued is mapped to.
              parameterValue:
                type: string
                description: The value put into the runbook for the parameter. Depending on the 'mappingType'
this value is optional (useDefault), represents the name of an incident parameter (parameter) or is the value
to be put in (fixed).

    Seasonality:
      type: object
      properties:
        policyId:
          type: string
          description: >
            The id of the policy that relates to the seasonality insight.
        seasonalDetails:
          type: object
          description: >
            Details or links on seasonality and/or to explain the reasons for
            the grouping.
    SimiliarIncident:
      type: object
      properties:
        incidentid:
          type: string
          description: id of incident that it related to
        link:
          type: string
          description: The URL to the incident in source system.
        distance:
          description: this should be a double but open API does not seem to like it  :-(
          type: integer

    SimilarIncidents:
      description: >
        Represents a list of similar historical tickets, based on query
        submitted to the service. It also includes a list of recommended
        actions that are extracted from resolutions, which can be highlighted
        with help of their offset information.
      type: object
      properties:
        similar_incidents:
          description: The list of similar incients for the query.
          type: array
          items:
            $ref: "./similar_incidents.schema.yaml#/components/schemas/SimilarIncidentItem"
        recommended_actions:
          description: The list of extracted actions from resolutions in related incidents.
          type: array
          items:
            $ref: "./similar_incidents.schema.yaml#/components/schemas/RecommendedActionItem"

    AlertResolutions:
      description: Represents a list of resolution links for current alert.
      type: object
      properties:
        shortDescription:
          type: string
          description: Short description of the log anomaly alert.
        description:
          type: string
```

```
          description: Description about the current log anomaly alert.
        category:
          type: string
          description: Detected error category.
        subCategory:
          type: string
          description: Detected error sub category.
        resolutionLinks:
          type: array
          description: A list of resolution url links for current alert.
          items:
            type: object
            properties:
              url:
                type: string
                description: URL link for the resolution.
              title:
                type: string
                description: Title for the resolution.
              content:
                type: string
                description: Content excerpt for the resolution.

    AssociatedAlertTemplates:
      description: >
        Represents the error template info for explainable purpose.
      type: object
      properties:
        error_template_list:
          type: array
          items:
            type: string
          description: A list of error template
        error_count_vector:
          type: array
          items:
            type: integer
          description: Count vector corresponding to error template list.
        significant_message_code:
          type: string
          description: Message code for the significant error template.
        type_window:
          type: string
          description: Indicates the window type, e.g., "liberty_twas"

    TopologicalGroup:
      description: >
        Signifies that one of the resources that this entity pertains to is a member
        of a resource group. The id of the insight is the id of the resource group
        within the topology store.
      type: object
      properties:
        name:
          type: string
          description: Human readable name of the resource group
        tags:
          description: A list of any tags assigned to the resource group
          type: array
          items:
            type: string
        entityTypes:
          description: >
            An array of group types that pertain to this group, ex. waiopsApplication which
            signifies this resource group represents an application.
          type: array
          items:
            type: string

    TopologicalResource:
      description: >
        Signifies that this entity pertains to a specific resource within the topology.
        The id of this insight will be set to the unique id of the resource within the topology.
      type: object
      properties:
        name:
          type: string
          description: Human readable name of the resource
        compositeOfIds:
          type: string
          description: >
            If this resource is a composite resource, contains a comma
```

```
            separated list of id's of composed resources.
        compositeId:
          type: string
          description: id of the compositeId

    ProbableCauseRanking:
      type: object
      description: the ranking of alerts in the group
      properties:
        id:
          type: string
          description: in the alert it is the correlationId or the eventId if in the incident
        rank:
          type: integer
          description: the rank of the alert

    IncidentTopology:
      type: object
      description: >
        details of the group id where incident topology is stored.  The incident topology is
        generated by probable cause component
      properties:
        id:
          type: string
          description: asm group id
        time:
          type: integer
          description: time in ms when the last resource in the asm group was added

    Union:
      type: object
      description: >
        Represents a union of insights of a particular type across entities. The
        union is uniquely identified by the id field of the insight.
      properties:
        subsumedUnions:
          type: array
          description: >
            An array of ids of previous union insights which were subsumed by
            this one, due to a new entity occurring that joined them.
          items:
            type: string
            description: The id of a subsumed union insight

    ProposedIncident:
      type: object
      description: >
        Represent a suggested incident to create for this alert.
      properties:
        policyId:
          type: string
          description: The id of the policy that proposes this incident.
        delay:
          type: integer
          description: Suggested hold-off delay in seconds before creating the incident.

    ChatOpsIntegrators:
      type: object
      description: >
        An array of configured integrators for a given incident to hold integrator specific details.
      properties:
        id:
          type: string
          description: Unique id for this integrator
        channel_name:
          type: string
          description: Chatops channel name
        name:
          type: string
          description: kind of chat ops i.e. slack, teams etc
        app_state:
          type: string
          description: Custom state field
        permalink:
          type: string
          description: Permalink for Integrators

    ITSMIntegrators:
      type: object
      description: >
        An array of configured ITSMs for a given incident to hold integrator specific details.
```

```
      properties:
        id:
          type: string
          description: Unique id for this platform
        ticket_num:
          type: string
          description: Ticket number associated with the ITSM system
        name:
          type: string
          description: kind of ITSM i.e. ServiceNow, etc
        app_state:
          type: string
          description: Custom state field
        permalink:
          type: string
          description: Permalink for ITSM platform
        type:
          type: string
          description: type of ITSM Integrators

    ActionHistory:
      description: >
        Represents the actions this entity has passed through.
      type: object
      properties:
        policyId:
          type: string
          description: The id of the policy that triggered the action
        actionId:
          type: string
          description: The id of the action the entity has passed through

    InsightDetails:
      anyOf:
        - $ref: "#/components/schemas/AssociatedChange"
        - $ref: "#/components/schemas/Conversation"
        - $ref: "#/components/schemas/Grouping"
        - $ref: "#/components/schemas/Runbook"
        - $ref: "#/components/schemas/Seasonality"
        - $ref: "#/components/schemas/SimiliarIncident"
        - $ref: "#/components/schemas/SimilarIncidents"
        - $ref: "#/components/schemas/AlertResolutions"
        - $ref: "#/components/schemas/AssociatedAlertTemplates"
        - $ref: "#/components/schemas/TopologicalGroup"
        - $ref: "#/components/schemas/TopologicalResource"
        - $ref: "#/components/schemas/ProbableCauseRanking"
        - $ref: "#/components/schemas/IncidentTopology"
        - $ref: "#/components/schemas/Union"
        - $ref: "#/components/schemas/ChatOpsIntegrators"
        - $ref: "#/components/schemas/ITSMIntegrators"
        - $ref: "#/components/schemas/ActionHistory"
```

Links property schema

```
Link:
      description: >
        Represents a link to an external system which has further information
        or control over an event.
      type: object
      properties:
        linkType:
          description: >
            Describes what type of link this is, giving a hint as to what may
            consume it.
          type: string
          example: webpage
        name:
          description: >
            An identifier for this link that can be referred to programmatically
          type: string
          example: management-console
        description:
          description: Descriptive text which defines what the link leads to
          type: string
          example: Fan controller management console
        url:
          description: >
            A fully qualified URL that can be used to locate the target of the
            link
          type: string
          example: https://fan-controller.example.com/?fanId=1234
```

Resource property schema

```
Resource:
      description: >
        Identifies a resource within the environment. This is made up of
        multiple fields that describe the resource. There are a set of standard
        fields which should be used where applicable, but additional fields may
        be set.
      type: object
      additionalProperties:
        description: Additional event-specific resource locators
        $ref: '#/components/schemas/StringOrInteger'
      properties:
        type:
          description: The type of resource
          type: string
          example: host
        name:
          description: The name of the resource
          type: string
          example: myhost.example.com
        sourceId:
          type: string
          description: The id the resource is known by in the source system
        hostname:
          description: The hostname of the resource
          type: string
          example: myhost.example.com
        ipAddress:
          description: The IP address of the resource
          type: string
          example: 9.123.123.1
        service:
          description: The name of the service that the resource is serving
          type: string
          example: mobile-app
        port:
          description: The port that is the subject of this event
          $ref: '#/components/schemas/StringOrInteger'
          example: "80"
        interface:
          description: The interface that is the subject of this event
          type: string
          example: eth0
        application:
          type: string
          description: The application that is the subject of this event
        controller:
          type: string
          description: The controller that is the subject of this event
        component:
          type: string
          description: The component that is the subject of this event
        cluster:
          type: string
          description: The cluster that is the subject of this event
        location:
          description: The location of the resource
          type: string
          example: Dallas 10
        accessScope:
          type: string
          description: The project or namespace the resource is part of
          example: default
        connectionId:
          type: string
          description: >-
            Configuration unique identifier of the connection this event
            came from.
          example: 'f5aa7fa9-92eb-4bec-942c-37eb3e3e9601'
```

Resource schemas

For more information about IBM Cloud Pak® for AIOps resource properties, see the following table.

Table. Resource schemas

Cloud Pak for AIOps resource property	Description	Type	Example
type	The type of the resource	string	host
name	The name of the resource	string	myhost.example.com
sourceId	The ID that the resource is known by in the source system	string	`7045b61d-e59c-4270-b525-545d55ecd06e`
hostname	The hostname of the resource	string	myhost.example.com
ipAddress	The IP address of the resource	string	9.123.123.1
service	The name of the service that the resource is serving	string	mobile-app
port	The port that is the subject of this event	string or integer	`ssh` or `22`
interface	The interface that is the subject of this event	string	eth0
application	The application that is the subject of this event	string	mobile-app-1
controller	The controller that is the subject of this event	string	controller1
component	The component that is the subject of this event	string	database
cluster	The cluster that is the subject of this event	string	staging-cluster
location	The location of the resource	string	Dallas 10
accessScope	The project or namespace that the resource is part of	string	default
connectionId	The unique identifier of the integration that this event came from	string	`f5aa7fa9-92eb-4bec-942c-37eb3e3e9601`
scopeId	The scope identifier value for this event from the originating system	string	myhost.example.com:location01

You can also define more resource properties for resources that are not included in the table. Added properties with valid JSON are stored and available, but special knowledge or handling of them is not included.

EventType schemas

EventType schemas indicate whether an event is a problem or a resolution and how it is classified. For more information about IBM Cloud Pak® for AIOps EventType properties, see the following table.

Cloud Pak for AIOps resource property	Description	Type	Example
eventType	Describes whether the event is a problem or resolution event. The only possible values that can be set are `problem` or `resolution`.	string	`problem`, `resolution`
classification	Describes the type of the event. This property is used to determine the golden signals for an alert.	string	`system status`, `threshold breach`, `utilization`
condition	Provides context to the type of the event and states the condition, status, or threshold that caused the event.	string	`> 30 seconds`, `95%`, `Down`, `Unavailable`

Normalized Mapping Rules

To help you map event data from incoming data sources review the following event schemas and samples.

- Kafka Connection Normalized Event schema
  - Normalized PagerDuty events sample
  - Normalized Netcool Operations Insight events sample
- PagerDuty Normalized Mapping Rules
  - Sample PagerDuty event
- Netcool Operations Insight normalized mapping rules
  - Sample Netcool Operations Insight event

Kafka Connection Normalized Event schema

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
"title": "Kafka Connection Normalized Event schema",
"description": "Normalized Event for None mapping use case",
"type": "object",
"properties": {
   "sender": {
```

```
   "description": "Identifies the resource which sent this event. This may be the resource that is the
subject of the event, or it may be some external system which is monitoring the subject.",
   "type": "object",
   "properties": {
      "service": {
         "description": "The identifier of the service",
         "type": "string"
      },
      "name": {
         "description": "The descriptive name of the service",
         "type": "string"
      },
      "type": {
         "description": "Data source name. For examples, PagerDuty, Netcool/OMNIbus",
         "type": "string"
      }
   },
   "required": [
      "service",
      "name",
      "type"
   ]
   },
   "resource": {
   "description": "Identifies the resource that is the subject of this event. This is made up of multiple
fields that describe the resource. There are a set of standard fields which should be used where applicable,
but additional fields may be set.",
   "type": "object",
   "properties": {
      "name": {
         "description": "The name of the resource. For example, myhost.example.com",
         "type": "string"
      },
      "hostname": {
         "description": "The hostname of the resource.",
         "type": "string"
      },
      "type": {
         "description": "The type of resource. For example, host",
         "type": "string"
      },
      "ipaddress": {
         "description": "The IP address of the resource.",
         "type": "string"
      },
      "location": {
         "description": "The location of the resource. For example, Dallas 10",
         "type": "string"
      }
   },
   "required": [
      "name",
      "hostname",
      "type",
      "ipaddress",
      "location"
   ]
   },
   "type": {
   "description": "Indicates whether the source alert is a problem or a resolution along with its
classification.",
   "type": "object",
   "properties": {
      "classification": {
         "description": "Description of the type of the event. E.g. Utilization, System status, Threshold
breach",
         "type": "string"
      },
      "eventType": {
         "description": "String description of whether the event is 'problem' or 'resolution'. Note: Only
these two values are permitted.",
         "type": "string"
      }
   },
   "required": [
      "classification",
      "eventType"
   ]
   },
   "severity": {
   "description": "Indicates the event severity level, which indicates how the perceived capability of the
```

```
managed object has been affected.",
   "type": "integer",
   "minimum": 0,
   "maximum": 6
   },
   "summary": {
   "description": "A human-readable description of the event.",
   "type": "string"
   },
   "occurrenceTime": {
   "description": "The time at which this event occurred. (ISO format: yyyy-mm-ddThh:mm:ss.SSSZ)",
   "type": "string",
   "pattern": "\\d{4}-[01]\\d-[0-3]\\dT[0-2]\\d:[0-5]\\d:[0-5]\\d(?:\\.\\d+)Z"
   },
   "expirySeconds": {
   "description": "The number of seconds before the event should automatically expire. A value of zero
indicates that there is no expiration.",
   "type": "integer",
   "minimum": 0
   }
},
"required": [
   "sender",
   "resource",
   "type",
   "severity",
   "summary",
   "occurrenceTime",
   "expirySeconds"
]
}
}
```

Normalized PagerDuty events sample

```
{
   "sender":{
      "service": "PEPF123",
      "name": "WAS_Sysdig",
      "type": "PagerDuty"
   },
   "resource":{
      "name":"Hostdown",
      "hostname":"alerts.sample.com",
      "type":"host",
      "ipaddress":"10.123.123.123",
      "location":"sample.mail.ibm.com"
   },
   "type":{
      "classification":"Pods ready less than desired is Triggered on kubernetes.cluster.name = RHOpenShift
and kubernetes.daemonSet.name = sysdig-agent",
      "eventType":"problem"
   },
   "severity":5,
   "summary":"Pods ready less than desired is Triggered on kubernetes.cluster.name = RHOpenShift and
kubernetes.daemonSet.name = sysdig-agent",
   "occurrenceTime":"2021-10-07T07:55:46.000Z",
   "expirySeconds":0
}
```

Normalized Netcool Operations Insight events sample

```
{
   "sender":{
      "service":"CEA SelfMonitoring",
      "name":"CEA SelfMonitoring @AGG_P",
      "type":"Netcool/OMNIbus"
   },
   "resource":{
      "name":"AGG_P",
      "hostname":"samplehost.ibm.com",
      "type":"host",
      "ipaddress":"10.1.96.24",
      "location":"Tokyo"
   },
   "type":{
      "classification":":++:CEASelfMonitoring",
      "eventType":"problem"
   },
```

```
   "severity":4,
   "summary":"CEAAnalytics SelfMonitoring Alert",
   "occurrenceTime":"2021-10-05T14:50:38.000Z",
   "expirySeconds":0
}
```

PagerDuty Normalized Mapping Rules

Note: Rules take precedence, that is the mapped value is taken from the highest rules that fulfils the condition.

Table. PagerDuty Normalized Mapping Rules

Resultant Normalized Alert			Mapping rules
{			
	"sender": {		
		"service": "P3ZB24W",	{{**service.id**}}
		"name": "PagerDuty - [SI] Beta - Website - Degraded",	"PagerDuty -" + {{**service.summary**}}
		"type": "PagerDuty"	"PagerDuty"
	},		
	"resource": {		
		"name": "Beta URL Down",	Rule 1: {{**body.cef_details.source_component**}}
			Rule 2: {{**incident.summary**}}
		"hostname": "alerts.newrelic.com",	Rule 1: hostname of {{**body.cef_details.details.incident_url**}}
			Rule 2: hostname of {{**incident.self**}}
		"type": "host",	Rule 1: "host" if the "**hostname**" field (see above) is hostname.
			Rule 2: "ip" if the "**hostname**" field (see above) is IPv4.
			Rule 3: Otherwise, empty.
		"ipaddress": "",	Rule 1: Same value as the "**hostname**" field if the value is IPv4
			Rule 2: Otherwise, empty.
		"location": "Beta URL GUI"	Rule 1: {{**body.cef_details.source_origin**}}
			Rule 2: {{**incident.summary**}}
	},		
	"type": {		
		"classification": "Beta URL Down (Beta URL GUI violated Beta URL Down)",	{{**summary**}}
		"eventType": "resolution"	Rule 1: "resolved" if {{**status**}} = "resolved"
			Rule 2: Otherwise "problem".
	},		
	"severity": 5,		{{**severity**}} -> AIOps severity level info -> 1
			low -> 2
			warning -> 3
			error -> 4
			critical -> 5
			Otherwise, 1
	"summary": "Beta URL Down (Beta URL GUI violated Beta URL Down)",		{{**summary**}}
	"occurrenceTime": "2021-07-11T03:39:53.000Z",		{{**created_at**}} converted to yyyy-mm-ddT:hh:mm:ss.000Z
	"expirySeconds": 0		0

Resultant Normalized Alert			Mapping rules
}			

Sample PagerDuty event used in the mapping


```
{
  "id": "Q2BSJKUORBWAWT",
  "type": "alert",
  "summary": "Beta URL Down (Beta URL GUI violated Beta URL Down)",
  "self": "https://api.pagerduty.com/alerts/Q2BSJKUORBWAWT",
  "html_url": "https://ibm-sandbox.pagerduty.com/alerts/Q2BSJKUORBWAWT",
  "created_at": "2021-07-10T20:39:53-07:00",
  "status": "resolved",
  "resolved_at": "2021-07-10T21:41:18-07:00",
  "alert_key": "799c00fbcf461bf553f9b4f30bcc1d9908db947c",
  "suppressed": false,
  "service": {
    "id": "P3ZB24W",
    "type": "service_reference",
    "summary": "[SI] Beta - Website - Degraded",
    "self": "https://api.pagerduty.com/services/P3ZB24W",
    "html_url": "https://ibm-sandbox.pagerduty.com/service-directory/P3ZB24W"
  },
  "severity": "critical",
  "incident": {
    "id": "P3HUMBH",
    "type": "incident_reference",
    "summary": "[#42113] Beta URL Down (Beta URL GUI violated Beta URL Down)",
    "self": "https://api.pagerduty.com/incidents/P3HUMBH",
    "html_url": "https://ibm-sandbox.pagerduty.com/incidents/P3HUMBH"
  },
  "first_trigger_log_entry": {
    "id": "R4Z5TVA4KYEVQL6DIFD9SHYPOY",
    "type": "trigger_log_entry_reference",
    "summary": "Triggered through the API",
    "self": "https://api.pagerduty.com/log_entries/R4Z5TVA4KYEVQL6DIFD9SHYPOY",
    "html_url": "https://ibm-
sandbox.pagerduty.com/alerts/Q2BSJKUORBWAWT/log_entries/R4Z5TVA4KYEVQL6DIFD9SHYPOY"
  },
  "body": {
    "contexts": [],
    "details": {
      "VIOLATION DESCRIPTION": null,
      "account_id": 2848358,
      "account_name": "Storage Insights",
      "closed_violations_count": {
        "critical": 0,
        "warning": 0
      },
      "condition_family_id": 15905725,
      "condition_id": 114530789,
      "condition_name": "Beta URL Down",
      "current_state": "open",
      "details": "2 or more locations failed on 'Beta URL GUI'",
      "duration": 65,
      "event_type": "INCIDENT",
      "incident_acknowledge_url":
"https://alerts.newrelic.com/accounts/2848358/incidents/313256636/acknowledge",
      "incident_id": 313256636,
      "incident_url": "https://alerts.newrelic.com/accounts/2848358/incidents/313256636",
      "metadata": {
        "entity.name": "Beta URL GUI",
        "entity.type": "MONITOR",
        "location_statuses": [
          {
            "location": "AWS_EU_CENTRAL_1",
            "locationLabel": "Frankfurt, DE",
            "result": "FAILED",
            "timestamp": 1625974724220
          },
          {
            "location": "AWS_US_WEST_1",
            "locationLabel": "San Francisco, CA, USA",
            "result": "FAILED",
            "timestamp": 1625974281070
          },
          {
```

```
            "location": "AWS_SA_EAST_1",
            "locationLabel": "São Paulo, BR",
            "result": "FAILED",
            "timestamp": 1625974154020
          },
          {
            "location": "AWS_AP_SOUTH_1",
            "locationLabel": "Mumbai, IN",
            "result": "FAILED",
            "timestamp": 1625974214555
          }
        ]
      },
      "open_violations_count": {
        "critical": 1,
        "warning": 0
      },
      "owner": "",
      "policy_name": "Beta URL Down",
      "policy_url": "https://alerts.newrelic.com/accounts/2848358/policies/1029635",
      "severity": "CRITICAL",
      "targets": [
        {
          "id": "03e19e74-cf92-4ea5-a0db-2e6d968fc9f9",
          "labels": {
            "account": "Storage Insights",
            "accountId": "2848358",
            "guid": "Mjg0ODM1OHxTWU5USHxNT05JVE9SfDAzZTE5ZTc0LWNmOTItNGVhNS1hMGRiLTJlNmQ5NjhmYzlmOQ",
            "monitorStatus": "Enabled",
            "monitorType": "Simple Browser",
            "period": "10",
            "publicLocation": "São Paulo, BR",
            "trustedAccountId": "1592783"
          },
          "link": "https://synthetics.newrelic.com/accounts/2848358/monitors/03e19e74-cf92-4ea5-a0db-
2e6d968fc9f9",
          "name": "Beta URL GUI",
          "product": "SYNTHETICS",
          "type": "Monitor"
        }
      ],
      "timestamp": 1625974792836,
      "timestamp_utc_string": "2021-07-11, 03:39 UTC",
      "version": "1.0",
      "violation_callback_url": "https://synthetics.newrelic.com/accounts/2848358/monitors/03e19e74-cf92-
4ea5-a0db-2e6d968fc9f9"
    },
    "cef_details": {
      "client": "New Relic",
      "client_url": "https://alerts.newrelic.com/accounts/2848358/incidents/313256636",
      "creation_time": "2021-07-11T03:39:53.673Z",
      "dedup_key": "799c00fbcf461bf553f9b4f30bcc1d9908db947c",
      "description": "Beta URL Down (Beta URL GUI violated Beta URL Down)",
      "details": {
        "VIOLATION DESCRIPTION": null,
        "account_id": 2848358,
        "account_name": "Storage Insights",
        "closed_violations_count": {
          "critical": 0,
          "warning": 0
        },
        "condition_family_id": 15905725,
        "condition_id": 114530789,
        "condition_name": "Beta URL Down",
        "current_state": "open",
        "details": "2 or more locations failed on 'Beta URL GUI'",
        "duration": 65,
        "event_type": "INCIDENT",
        "incident_acknowledge_url":
"https://alerts.newrelic.com/accounts/2848358/incidents/313256636/acknowledge",
        "incident_id": 313256636,
        "incident_url": "https://alerts.newrelic.com/accounts/2848358/incidents/313256636",
        "metadata": {
          "entity.name": "Beta URL GUI",
          "entity.type": "MONITOR",
          "location_statuses": [
            {
              "location": "AWS_EU_CENTRAL_1",
              "locationLabel": "Frankfurt, DE",
              "result": "FAILED",
              "timestamp": 1625974724220
```

```
            },
            {
              "location": "AWS_US_WEST_1",
              "locationLabel": "San Francisco, CA, USA",
              "result": "FAILED",
              "timestamp": 1625974281070
            },
            {
              "location": "AWS_SA_EAST_1",
              "locationLabel": "São Paulo, BR",
              "result": "FAILED",
              "timestamp": 1625974154020
            },
            {
              "location": "AWS_AP_SOUTH_1",
              "locationLabel": "Mumbai, IN",
              "result": "FAILED",
              "timestamp": 1625974214555
            }
          ]
        },
        "open_violations_count": {
          "critical": 1,
          "warning": 0
        },
        "owner": "",
        "policy_name": "Beta URL Down",
        "policy_url": "https://alerts.newrelic.com/accounts/2848358/policies/1029635",
        "severity": "CRITICAL",
        "targets": [
          {
            "id": "03e19e74-cf92-4ea5-a0db-2e6d968fc9f9",
            "labels": {
              "account": "Storage Insights",
              "accountId": "2848358",
              "guid": "Mjg0ODM1OHxTWU5USHxNT05JVE9SfDAzZTE5ZTc0LWNmOTItNGVhNS1hMGRiLTJlNmQ5NjhmYzlmOQ",
              "monitorStatus": "Enabled",
              "monitorType": "Simple Browser",
              "period": "10",
              "publicLocation": "São Paulo, BR",
              "trustedAccountId": "1592783"
            },
            "link": "https://synthetics.newrelic.com/accounts/2848358/monitors/03e19e74-cf92-4ea5-a0db-
2e6d968fc9f9",
            "name": "Beta URL GUI",
            "product": "SYNTHETICS",
            "type": "Monitor"
          }
        ],
        "timestamp": 1625974792836,
        "timestamp_utc_string": "2021-07-11, 03:39 UTC",
        "version": "1.0",
        "violation_callback_url": "https://synthetics.newrelic.com/accounts/2848358/monitors/03e19e74-cf92-
4ea5-a0db-2e6d968fc9f9"
      },
      "message": "Beta URL Down (Beta URL GUI violated Beta URL Down)",
      "mutations": [],
      "reporter_component": "08fe44fc923247aa9b931d186559a105",
      "severity": "critical",
      "source_component": "Beta URL Down",
      "source_origin": "Beta URL GUI",
      "version": "1.0"
    },
    "type": "alert_body"
  },
  "integration": {
    "id": "PL68E48",
    "type": "generic_events_api_inbound_integration_reference",
    "summary": "New Relic",
    "self": "https://api.pagerduty.com/services/P3ZB24W/integrations/PL68E48",
    "html_url": "https://ibm-sandbox.pagerduty.com/services/P3ZB24W/integrations/PL68E48"
  },
  "privilege": null
}
```


Netcool Operations Insight normalized mapping rules

Note: Rules take precedence, that is the mapped value is taken from the highest rules that fulfils the condition.

Table. Netcool Operations Insight normalized mapping rules

Resultant Normalized Alert			Mapping rules
{			
	"sender": {		
		"service": "Ping Probe",	{{**Agent**}}
		"name": "Ping Probe",	{{**Manager**}}
		"type": "Netcool/OMNIbus"	"Netcool/OMNIbus"
	},		
	"resource": {		
		"name": "RNIB-MPLS-FTTC013",	Rule 1: Take {{**Node**}} if it is not empty
			Rule 2: Take {{**NodeAlias**}} if it is not empty
			Rule 3: Rule 3: Otherwise "unknown"
		"hostname": "RNIB-MPLS-FTTC013",	Rule 1: {{**Node**}} if not empty and is valid hostname
			Rule 2: {{**NodeAlias**}} if not empty and is valid hostname
		"type": "host",	Rule 1: "host" if either {{**Node**}} or {{**NodeAlias**}} is not empty,
			Rule 2: Otherwise, "unknown".
		"ipaddress"10.44.113.88",	Rule 1: {{**Node**}} if not empty and is valid IPv4
			Rule 2: {{**NodeAlias**}} if not empty and is valid IPv4
			Rule 3: Otherwise, empty
		"location": "Paper St"	{{**Location**}}
	},		
	"type": {		
		"classification": "probeping-icmptimeout:++:Ping Status",	{{**EventId**}} ":++:" {{**AlertGroup**}}
		"eventType": "problem"	Rule 1: "resolution" if {{**Severity**}} is 0.
			Rule 2: "resolution" if {{**Type**}} is one of {2,4}
			Rule 3: Otherwise, "problem"
	},		
	"severity": 3,		Rule 1: {{**Severity**}} + 1 if {{**Severity**}} falls within 2 to 5.
			Rule 2: 1 if {{**Severity**}} is 1
			Rule 3: 2 if {{**Severity**}} is > 0
	"summary": "RNIB-MPLS-FTTC013 is not reachable",		{{**Summary**}}
	"occurrenceTime": "2021-02-23T09:56:46.000Z",		Output format: yyyy-mm-ddThh:mm:ss.SSSZ
			Rule 1: If {{**NetcoolEventAction**}} is "INSERT", then {{**FirstOccurrence**}} - yyyy-mm-ddThh:mm:ss {{**FirstOccurrenceUSec**}} - if available then .SSSZ, otherwise .000Z
			Rule 2: If {{**NetcoolEventAction**}} is "UPDATE", then {{**LastOccurrence**}} - yyyy-mm-ddThh:mm:ss {{**LastOccurrenceUSec**}} - if available then .SSSZ, otherwise .000Z
			Note: {{**FirstOccurrence**}} and {{**LastOccurrence**}} can be in Unix epoch or datetime string.
	"expirySeconds": 0		Rule 1: Take {{**ExpireTime**}} if the field exists.
			Rule 2: Otherwise, 0.
}			

Sample Netcool Operations Insight event used in the mapping

```
  {
    "Identifier": "RNIB-MPLS-FTTC013 Ping Status:10.44.113.88 Ping Status 1 Ping Probe Ping Probe RNIB-MPLS-FTTC013 unreachable",
```

```
    "SERIAL": 25230712,
    "Node": "RNIB-MPLS-FTTC013",
    "NodeAlias": "10.44.113.88",
    "Manager": "Ping Probe",
    "Agent": "Ping Probe",
    "AlertGroup": "Ping Status",
    "AlertKey": "Ping Status:10.44.113.88",
    "Severity": 3,
    "Summary": "RNIB-MPLS-FTTC013 is not reachable",
    "StateChange": "1614074206",
    "FirstOccurrence": "1614074206",
    "LastOccurrence": "1614074206",
    "LASTMODIFIED": "1614074206",
    "Poll": 0,
    "Type": 1,
    "Tally": 1,
    "Class": 100,
    "Grade": 0,
    "Location": " Paper St",
    "OwnerUID": 65534,
    "OwnerGID": 0,
    "Acknowledged": 0,
    "Flash": 0,
    "EventId": "probeping-icmptimeout",
    "ExpireTime": 0,
    "ProcessReq": 0,
    "SuppressEscl": 0,
    "Customer": "RNIB",
    "Service": "",
    "PhysicalSlot": 0,
    "PhysicalPort": 0,
    "PhysicalCard": "",
    "TaskList": 0,
    "NmosSerial": "",
    "NmosObjInst": 0,
    "NmosCauseType": 0,
    "LocalNodeAlias": "10.44.113.88",
    "LocalPriObj": "",
    "LocalSecObj": "",
    "LocalRootObj": "",
    "RemoteNodeAlias": "",
    "RemotePriObj": "",
    "RemoteSecObj": "",
    "RemoteRootObj": "",
    "X733EventType": 0,
    "X733ProbableCause": 0,
    "X733SpecificProb": "",
    "X733CorrNotif": "",
    "ServerName": " AGG_P",
    "ServerSerial": 25230712,
    "DELETEDAT": "2015-01-01T01:06:22",
    "ACKOWNERUID": 0,
    "ALERTCATEGORY": "0",
    "CIRCUIT": "BBEU07581255",
    "DESCRIPTION": "A device is not responding to ICMP ping monitoring",
    "EVENTCATEGORY": "1",
    "EVENTOWNERGID": 0,
    "EVENTSUBCATEGORYONE": "1",
    "EVENTSUBCATEGORYTWO": "1",
    "ITMDISPLAYITEM": "",
    "ITMSITFORMULA": "",
    "ITMSITTYPE": "S",
    "LINEID": "",
    "PARTYID": "0",
    "TICKETID": 0,
    "TIMEESCL": "1970-01-01T01:00:00",
    "TIMEFIRSTACK": "1970-01-01T01:00:00",
    "TIMEJOURN": "1970-01-01T01:00:00",
    "TIMELASTACK": "1970-01-01T01:00:00",
    "TIMETICKET": "1970-01-01T01:00:00",
    "CUSTOMERTIER": "0",
    "CUSTOMERSCORE": "0",
    "EVENTPRIORITY": "93",
    "EVENTORIGIN": "1",
    "TIMECLEAR": "2015-01-01T00:05:15",
    "CIRCUITTYPE": "ADSL",
    "SNDCCLASSNAME": "cmdb_ci_ip_router",
    "SNDCMESSINASHORTNAME": "RNIBA",
    "SNDCSYSID": "de16ced88712e004055fe5b8c9434d54",
    "MANUFACTURER": "Cisco",
    "CUSTOMERALIASNAME": "Paper St - ADSL [CIRCUITID:MKVL435293]",
```

```
    "SNDCMESSINAOAID": "1191313",
    "SNDCACCESSROUTER": "",
    "SNDCACCESSROUTERPORT": "",
    "SNDCBEARER": "",
    "SNDCINSTALLNUMBER": "01509269339 ",
    "SNDCDSLCARETYPE": "Enhanced",
    "TICKETREF": "",
    "TELCO": "ACME Telecom",
    "SERVICEAFFECTING": "0",
    "CUSTOMERREFERENCE": "UK-RNIBA",
    "TIMEEVENTOWNERGID": "1970-01-01T01:00:00",
    "TICKETREFLAST": "",
    "SUPPORTSEGMENT": "2",
    "NETWORKINTERFACE": "",
    "URL": "",
    "CORRFLAG": "0",
    "PARENTCHILD": "0",
    "PARENTSERIAL": 0,
    "NEWSUMMCOUNT": -1,
    "CMSURI": "",
    "ENTITYTYPE": "Circuit_ADSL",
    "NetcoolEventAction": "insert"
  }
```

References: Infrastructure Automation

View reference information about Infrastructure Automation.

- Infrastructure management
- Managed services

References: Infrastructure management

View reference information about Infrastructure Automation Infrastructure management.

- Installation parameters for the Managed services
- Support Matrix
- Deployment Planning Guide
- Integration with OpenShift Container Platform
- High Availability Guide
- Appliance Hardening Guide
- Methods Available for Automation
- Managing Authentication for Infrastructure Management (stand-alone)
- Scripting Actions in Infrastructure Management (stand-alone)
- General Configuration
- Self Service User Interface Guide

Installation parameters

The following table lists the Managed services installation parameters and their default values. These parameters are in the **spec** section of the **ibm-management-cam-install** operator.

Notes:

- The storage class **rook-ceph-cephfs-internal** is specified as an example in this documentation, use the storage class configured in your environment.
- The global resource parameters are used while creating the following containers:

cam-controller-manager, **cam-iaas**, **cam-portal-ui**, **cam-provider-ansible**, **cam-provider-bpm**, **cam-provider-helm**, **cam-provider-terraform-api**, **cam-provider-terraform-runtime**, **cam-proxy**, **cam-service-composer-api**, **cam-service-composer-ui**, **cam-tenant-api**, **cam-ui-basic**.

Table 1. Managed services parameters

Parameter	Description	Default

Parameter	Description	Default
Authentication		
`oauthClientSecret`	Name of the OAuth client secret	cam-oauth-client-secret
`authIdpCredentialsSecret`	Name of the secret with the auth IDP credentials	ibm-iam-bindinfo-platform-auth-idp-credentials
Architecture		
`arch`	Select the value based on the architecture. Linux = amd64 Linux on Power (ppc64le) = ppc64le Linux on IBM Z and LinuxONE = s390x	amd64
Audit service		
`auditService.config.journalPath`	The journal path	/run/systemd/journal
`auditService.image.pullPolicy`	Pull policy for the audit image	IfNotPresent
`auditService.image.pullSecret`	Image pull secret for the audit image	
`auditService.image.repository`	Repository for the audit image	quay.io/opencloudio/
`auditService.image.tag`	Image tag of the audit service	IBM Cloud Pak foundational services version
`auditService.resources.limits.cpu`	CPU resource limits	200m
`auditService.resources.limits.memory`	Memory resource limits	512Mi
`auditService.resources.requests.cpu`	CPU resource request	100m
`auditService.resources.requests.memory`	Memory resource request	256Mi
API		
`camAPI.camSecret.secretName`	Name of the secret used by the API	`cam-api-secret`
`camAPI.certificate.certName`	Name of the certificate used by the API	`cert`
`camAPI.replicaCount`	Number of API pods	`1`
Ansible provider		
`camAnsibleProvider.replicaCount`	Number of pods. Set to `1` if you want to use Ansible provider. After you install Managed services, to enable or disable Ansible integration, see Enable or disable Ansible Integration.	`0`
Cloud Discovery		
`camBPDCDS.options.customSettingsFile`	Cloud Discovery Service custom settings file	Nil
`camBPDCDS.options.debug.enabled`	Enable Cloud Discovery Service debug mode	`false`
`camBPDCDS.replicaCount`	Number of Cloud Discovery Service pods	`1`
`camBPDCDS.resources.limits.cpu`	CPU resource limits	`200m`
`camBPDCDS.resources.limits.memory`	Memory resource limits	`256Mi`
`camBPDCDS.resources.requests.cpu`	CPU resource requests	`100m`
`camBPDCDS.resources.requests.memory`	Memory resource requests	`128Mi`
Business Process Management		

Parameter	Description	Default
`camBpmProvider.replicaCount`	Number of pods. Set to `1` if you want to use IBM Business Process Manager. After you install Managed services, to enable or disable IBM Business Process Manager integration, see Enable or disable IBM Business Process Manager.	`0`
`camBpmProvider.bpmEndpoint`	IBM BPM endpoint	""
`camBpmProvider.bpmUserName`	IBM BPM user	""
`camBpmProvider.bpmUserPwd`	IBM BPM user password	""
Terraform		
`camProviderTerraform.isolateRuntime`	Set to `true` to enable isolation mode for Provider Terraform runtime. In this mode, the runtime runs on separate pods.	`false`
`camProviderTerraform.runtime.replicaCount`	Number of replicas for Provider Terraform Runtime. This value is applicable only when `camProviderTerraform.isolateRuntime` is enabled.	`0`
`camProviderTerraform.terraformSSHDir.terraformSSHDirSecret`	Secret that contains the contents of /home/terraform/.ssh directory. See Configuring Managed services to set the SSH keys and known hosts in provider terraform pod for more details.	""
Controller		
`camController.replicaCount`	Set this to 1 to install the CAM controller.	`1`
Logging policy		
`camLoggingPolicies.logLevel`	Logging level.	`info`
MongoDB PV		
`camMongoPV.name`	Name of MongoDB volume	`cam-mongo-pv`
`camMongoPV.persistence.accessMode`	PV access mode. Do not change the default value.	ReadWriteOnce
`camMongoPV.persistence.enabled`	Database persistence	`true`
`camMongoPV.persistence.existingClaimName`	Database PVC name	`nil`
`camMongoPV.persistence.existingDynamicVolume`	Whether to use an existing PV	`false`
`camMongoPV.persistence.size`	Database PV size	`15Gi`
`camMongoPV.persistence.storageClassName`	Database storage class	`rook-ceph-cephfs-internal`
`camMongoPV.persistence.useDynamicProvisioning`	The default is true. The IBM Cloud Pak for AIOps default storage class is used if you do not change the storage class in the `camLogsPV.persistence.storageClassName` parameter.	`true`
Proxy		
`camProxy.replicaCount`	Number of proxy pods	`1`
Managed services console		
`camUI.camUISecret.secretName`	Secret that is used by the UI	`cam-ui-secret`
`camUI.camUISecret.sessionKey`	Session key that is used by the UI	`opsConsole.sid`
`camUI.camUISecret.sessionTTL`	UI session expiry time	`43200` seconds
`camUI.replicaCount`	Number of UI pods	1
MongoDB setting - External or internal		
`database.bundled`	Use internal MongoDB. Set to false for external. For more information, see Using an external MongoDB	`true`
Global parameters		
`global.audit`	Controls whether the audit container is added to the build or not. It also enables or disables audit in the service.	false

Parameter	Description	Default
`global.iam.deployApiKey`	It is used to generate Managed services specific Service ID and API Key. The Managed services API Key is used to interact with the platform authorization. If this value is not specified, the installation generates it automatically.	
`global.id.productID`	Managed services Product Identifier	Defaults to the Managed services product ID. If you are downloading the package from IBM Passport Advantage, you can specify the part number here.
`global.id.productVersion`	IBM Cloud Pak for AIOps version	IBM Cloud Pak for AIOps version
`global.id.cloudpakID`	IBM Cloud Pak for AIOpsID	Defaults to the IBM Cloud Pak for AIOps product ID.
`global.id.cloudpakVersion`	IBM Cloud Pak for AIOps version	IBM Cloud Pak for AIOps version
`global.image.secretname`	Entitled registry image pull secret	Type `default` to use your namespace default service account image pull secret or enter your Entitled registry image pull secret name. Required if you are downloading images from the Entitled registry.
`global.offline`	Select to optimize the installation for offline. For example, skip attempting to access the internet for public starter templates.	false
Cluster parameters		
`icp.cluster_name`	Your IBM Cloud Pak for AIOps cluster name.	mycluster.
`icp.port`	Port for accessing IBM Cloud Pak for AIOps. The port is used to communicate with the services in IBM Cloud Pak for AIOps.	443
Image parameters		
`image.pullPolicy`	Pull Policy for the Managed services images	IfNotPresent
`image.repository`	Repository for the Managed services images	`quay.io/opencloudio/` (online installation image repository) `mycluster.icp:8500/services/` (offline installation image repository)
`image.tag`	Tag for the Managed services images	IBM Cloud Pak foundational services version
Managed services console hostname and port		
`managementConsole.hostname`	Hostname of the Managed services management console. If this field is kept blank, the installer generates a hostname. You can access the hostname by running the command `oc get route -n cp4aiops`.	NA
`managementConsole.port`	Port for Managed services management console. If this field is kept blank, a random valid port is chosen.	The default is 30000 and the valid range is 30000-32767.
Environment parameters		
`platform`	Platform name.	icp
Proxy parameters		
`proxy.useProxy`	Select to use a proxy with Managed services.	`false`
Global resource parameters		
`resources.limits.cpu`	CPU resource limits	`1`
`resources.limits.memory`	Memory resource limits	`8Gi`
`resources.requests.cpu`	CPU resource requests	`100m`
`resources.requests.memory`	Memory resource requests	`256Mi`
Custom secrets		
`secureValues.secretName`	Name of a secret that you created before the installation of Managed services. It contains secure values, such as proxy, database, and encryption passwords.	Nil. To generate the secret, see Using pre-created secrets in Managed services chart.
Red Hat® OpenShift® on IBM Cloud® service		
`roks`	Indicates if it is a Red Hat® OpenShift® on IBM Cloud® service system	False

Parameter	Description	Default
`roksRegion`	Red Hat® OpenShift® on IBM Cloud® service storage region	""
`roksZone`	Red Hat® OpenShift® on IBM Cloud® service storage zone	""

Supported platforms and features in Infrastructure management

Infrastructure management includes support for different platforms and features.

- Introduction
- Supported Browsers
- Supported Platforms
- Supported Providers
- Provisioning Support on Infrastructure Providers
- Provisioning Support on Cloud Providers
- Feature Support on Infrastructure Providers
- Feature Support on Cloud Providers
- Feature Support on Container Providers
- Feature Support on Network Providers
- Feature Support on Configuration Management Providers
- Remote Console Support
- Operating System Support
- Smart State Analysis Support
- Supported Guest Operating Systems

Chapter 1. Introduction

Infrastructure Management is also delivered as a virtual appliance that includes the following key components:

- CentOS Linux release (Core)
- Rails
- Ruby
- PostgreSQL
- Infrastructure Management Engine
- Apache

Chapter 2. Supported Browsers

The following table outlines the browsers that are supported for viewing the Infrastructure Management web user interface.

Browser support is divided into tiers:

- Tier 1: Browser and operating system combinations that are fully tested and fully supported. IBM is committed to fixing issues with browsers on this tier.
- Tier 2: Browser and operating system combinations that are partially tested, and are likely to work. Limited support is provided for this tier. IBM attempts to fix issues with browsers on this tier.
- Tier 3: Browser and operating system combinations that are not tested, but might work. Minimal support is provided for this tier. IBM attempts to fix only minor issues with browsers on this tier.

Table 2.1. Supported Browsers

Support tier	Browser
Tier 1	Google Chrome - Chrome for Business. See Chapter 6. Remote Console Support for information on VMRC plug-in support. Mozilla Firefox - Firefox Extended Support Release
Tier 2	Microsoft Internet Explorer Edge - version 44 and newer
Tier 3	Safari on MacOS (Mojave) - version 12.1.x

Chapter 3. Supported Platforms for Infrastructure management

The following table lists platforms that can host the Infrastructure management appliance, and platforms that can be managed by Infrastructure management as providers.

Table 3.1. Supported Platforms

Platform	Support for Infrastructure Management Appliance	Supported for Management
Amazon EC2 [a]	Yes	Yes
IBM Cloud	Yes	Yes
Microsoft Azure [a]	No	Yes
Google Compute Engine	No	Yes
Oracle Cloud	No	Yes
OpenShift Container Platform (OCP) 4.1, 4.2, 4.3, and 4.4	No	Yes
OpenShift Container Platform (OCP) 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, and 4.11	Yes	Yes
Red Hat OpenStack Platform (OSP) 10 [b]	Yes	Yes
Red Hat OpenStack Platform (OSP) 13 [b]	Yes	Yes
Red Hat OpenStack Platform (OSP) 16 [b]	Yes	Yes
Red Hat Virtualization (RHV) 4.2 [c]	Yes	Yes
Red Hat Virtualization (RHV) 4.3 [c]	Yes	Yes
Red Hat Virtualization (RHV) 4.4 [c]	Yes	Yes
Red Hat Virtualization (RHV) 4.4SP1 [c]	Yes	Yes
VMware vSphere 6.0	Yes	Yes
VMware vSphere 6.5	Yes	Yes
VMware vSphere 6.7	Yes	Yes
VMware vSphere 7	Yes	Yes
VMware vSphere 8	Yes	Yes

[a] As the pace of change in public cloud features is high, constantly changing and evolving, the compatibility for Infrastructure Management is as of the time of the GA of the product. Best effort basis during errata cycle for public cloud vendor introduced breaking changes (intended or unintended) to existing functionality. Any other changes or additions would be considered as a Request For Enhancement (RFE) which is evaluated for potential inclusion in future releases. RFE can be requested by way of your standard support mechanisms.

[b] See Red Hat OpenStack Platform Life Cycle for product lifecycle information.

[c] See Red Hat Enterprise Virtualization Life Cycle for product lifecycle information.

Chapter 4. Supported Providers

Provider support is divided into tiers:

- Tier 1: These providers are developed, tested, maintained, and supported by IBM. IBM accepts support and enhancement requests.
- Tier 2: These providers are delivered by an IBM partner. IBM accepts support and enhancement requests, but all support and enhancements updates are provided by the partner.
- Tier 3: These providers are delivered by various sources into the ManageIQ open source community. IBM packages and ships these providers and accepts support cases. All support and enhancement updates need to come from the open source community.

Table 4.1. Supported Providers

Support tier	Provider
Tier 1	Amazon, Amazon EKS, Azure, OpenStack, VMware vCenter, Google Cloud, IBM Cloud, Oracle Cloud, Ansible, OpenShift, RHEV, and Satellite, IBM Power HMC, IBM PowerVC, IBM Power Systems Virtual Server
Tier 2	IBM SDE Storage Automation
Tier 3	Microsoft System Center VMM (Hyper-V), OpenStack Platform Director, Azure Stack, Redfish, Nuage Network Manager, VMware vCloud, VMware NSX-t, Lenovo XClarity, and Kubernetes

Chapter 5. Provisioning Support

5.1. Infrastructure Providers

The following table outlines the status of support for provisioning on infrastructure providers.

Table 5.2. Infrastructure Provider Provisioning Support

Feature	Red Hat Virtualization (RHV) Manager	VMware vCenter	IBM Power HMC		
Provision VM/Instance using PXE	Yes	No	No		
Provision VM/Instance using ISO	Yes	No	No		

Feature	Red Hat Virtualization (RHV) Manager	VMware vCenter	IBM Power HMC		
Clone from VM/Instance to VM/Instance	No	Yes	No		
Provision from Template/Image to VM/Instance	Yes	Yes	Yes		
Provision from VM/Instance to Template/Image	No	Yes	Yes		

5.2. Cloud Providers

The following table outlines the status of support for provisioning on cloud providers.

Table 5.3. Cloud Provider Provisioning Support

Feature	OpenStack Platform (OSP)	Amazon EC2	Microsoft Azure	Google Compute Engine	IBM Cloud VPC	IBM PowerVS	IBM Power Systems Virtual Server
Provision VM/Instance using PXE	No	No	No	No	No	No	No
Provision VM/Instance using ISO	No	No	No	No	No	No	No
Clone from VM/Instance to VM/Instance	No	No	No	No	No	No	No
Provision from Template/Image to VM/Instance	Yes	Yes	Yes	Yes	No	Yes	Yes
Provision from VM/Instance to Template/Image	No	No	No	No	Yes	No	No
Orchestration Template Provisioning	Yes	Yes	Yes	No	No	No	No

Chapter 6. Feature Support

Infrastructure management allows you to perform actions on the providers that it manages, and provides features for interacting with and providing information about those providers. This section outlines the status of support for actions that you can perform on providers, and the Infrastructure management features that allow you to interact with those providers.

Note: You can run a rake command to retrieve a list of provider features supported across all platforms. Run the following command in your Infrastructure management appliance to retrieve the supported features in a comma-separated values (.csv) file format. The comma-separated values are displayed on the command-line interface (CLI) by default. You can use a pipe to redirect the output to a file.

```
ruby /var/www/miq/vmdb/tools/feature_support_matrix.rb
```

6.1. Infrastructure Providers

The following tables outline the status of support for Infrastructure management features on infrastructure providers.

6.1.1. Red Hat Virtualization

The following table outlines the status of support for Infrastructure management features on Red Hat Virtualization (RHV) infrastructure providers.

Table 6.1. Red Hat Virtualization Provider Feature Support

Feature	RHV 4.2	RHV 4.3	RHV 4.4
Relationship Discovery	Yes	Yes	Yes
VM Drift Comparison	Yes	Yes	Yes
Track VM Genealogy	Yes	Yes	Yes
Capacity & Utilization	Yes	Yes	Yes
Capture VM/Instance Event Timelines	Yes	Yes	Yes
Reporting	Yes	Yes	Yes

Feature	RHV 4.2	RHV 4.3	RHV 4.4
Right Sizing	Yes	Yes	Yes
Chargeback	Yes	Yes	Yes
Remote Console VM Access	Yes [a]	Yes [a]	Yes [a]
Snapshot Creation and Removal	Yes	Yes	Yes
VM / Instance Compliance Enforcement	Yes	Yes	Yes
VM / Instance Policy Enforcement	Yes	Yes	Yes
VM / Instance Power Operations	Yes	Yes	Yes
VM / Instance Retirement	Yes	Yes	Yes
Alerts - VM Value Changed	Yes	Yes	Yes
Alerts - VM Reconfigured	Yes	Yes	Yes
Integrate with Service Catalogs	Yes	Yes	Yes
Virtual Machine Reconfiguration	Yes	Yes	Yes
Automation Work Flows	Yes	Yes	Yes
Tag Mapping from Provider	No	No	No
Tag Mapping to Provider	No	No	No
Provision VM/Instance using PXE	Yes	Yes	Yes
Provision VM/Instance using ISO	Yes	Yes	Yes
Provision from Template/Image to VM/Instance	Yes	Yes	Yes
Provision from VM/Instance to Template/Image	Yes	Yes	Yes
Host Power Operations	No	No	No
Provision Host	No	No	No
Clone from VM/Instance to VM/Instance	No	No	No
Cloud-Init	Yes	Yes	Yes
Customize Windows Templates with Sysprep during Provisioning	Yes	Yes	Yes
OVN Provider	Yes	Yes [b]	Yes [b]

[a] On some operating system and browser combinations.

[b] Technology preview.

6.1.2. VMware vSphere

The following table outlines the status of support for Infrastructure management features on VMware vSphere infrastructure providers.

Table 6.2. VMware vSphere Provider Feature Support

Feature	vSphere 6.0	vSphere 6.5	vSphere 6.7	vSphere 7	vSphere 8
Relationship Discovery	Yes	Yes	Yes	Yes	Yes
VM Drift Comparison	Yes	Yes	Yes	Yes	Yes
Track VM Genealogy	Yes	Yes	Yes	Yes	Yes
Capacity & Utilization	Yes	Yes	Yes	Yes	Yes
Capture VM/Instance Event Timelines	Yes	Yes	Yes	Yes	Yes
Disk Addition to VM/Instance	Yes	Yes	Yes	Yes	Yes
Reporting	Yes	Yes	Yes	Yes	Yes
Right Sizing	Yes	Yes	Yes	Yes	Yes
Chargeback	Yes	Yes	Yes	Yes	Yes
Remote Console VM Access	Yes [a]	Yes [a]	Yes [a]	Yes [a]	Yes [a]
Snapshot Creation and Removal	Yes	Yes	Yes	Yes	Yes
VM / Instance Compliance Enforcement	Yes	Yes	Yes	Yes	Yes
VM / Instance Policy Enforcement	Yes	Yes	Yes	Yes	Yes
VM / Instance Power Operations	Yes	Yes	Yes	Yes	Yes
VM / Instance Retirement	Yes	Yes	Yes	Yes	Yes
Alerts - Real Time	Yes	Yes	Yes	Yes	Yes
Alerts - VM Value Changed	Yes	Yes	Yes	Yes	Yes
Alerts - VM Reconfigured	Yes	Yes	Yes	Yes	Yes
Integrate with Service Catalogs	Yes	Yes	Yes	Yes	Yes
Virtual Machine Reconfiguration	Yes	Yes	Yes	Yes	Yes
Automation Work Flows	Yes	Yes	Yes	Yes	Yes
Tag Mapping from Provider	No	No	No	No	No
Tag Mapping to Provider	No	No	No	No	No

Feature	vSphere 6.0	vSphere 6.5	vSphere 6.7	vSphere 7	vSphere 8
Provision VM/Instance using PXE	Yes	Yes	Yes	Yes	Yes
Provision VM/Instance using ISO	No	No	No	No	No
Provision from Template/Image to VM/Instance	Yes	Yes	Yes	Yes	Yes
Provision from VM/Instance to Template/Image	Yes	Yes	Yes	Yes	Yes
Host Power Operations	Yes	Yes	Yes	Yes	Yes
Customize Windows Templates with Sysprep during Provisioning	Yes	Yes	Yes	Yes	Yes
Clone from VM/Instance to VM/Instance	Yes	Yes	Yes	Yes	Yes
[a] On some operating system and browser combinations					

6.1.3. IBM Power HMC

The following table outlines the status of support for IBM Power HMC infrastructure providers.

Table 6.3. IBM Power HMC Provider Feature Support

Feature	IBM Power HMC
Relationship Discovery	Yes
VM Drift Comparison	No
Track VM Genealogy	No
Capacity & Utilization	Yes
Capture VM/Instance Event Timelines	Yes
Reporting	Yes
Right Sizing	Yes
Chargeback	No
Remote Console VM Access	No
Snapshot Creation and Removal	No
VM / Instance Compliance Enforcement	No
VM / Instance Policy Enforcement	No
VM / Instance Power Operations	Yes
VM / Instance Retirement	Yes
Alerts - Real Time	No
Alerts - VM Value Changed	No
Alerts - VM Reconfigured	No
Integrate with Service Catalogs	No
Virtual Machine Reconfiguration	No
Automation Work Flows	No
Tag Mapping from Provider	No
Tag Mapping to Provider	No
Provision VM/Instance using PXE	No
Provision VM/Instance using ISO	No
Provision from Template/Image to VM/Instance	Yes
Provision from VM/Instance to Template/Image	Yes
Host Power Operations	Yes
Provision Host	No
Clone from VM/Instance to VM/Instance	No

6.2. OpenStack Platform Director Infrastructure Providers

The following table outlines the status of support for Infrastructure management features on OpenStack Platform (OSP) director infrastructure providers.

Table 6.5. OpenStack Platform Director Infrastructure Provider Feature Support

Feature	OpenStack Platform (OSP) Director
Relationship Discovery	Yes
Nodes Inventory	Yes
OpenStack Services Inventory	Yes
Nodes Drift Comparison	Yes

Feature	OpenStack Platform (OSP) Director
Nodes Smart State	Yes
Capacity & Utilization	Yes
Capture Infrastructure Event Timelines	Yes
Node Power Operation	Yes
Capacity Planning	Yes
Reporting	Yes
Add/Remove Node	Yes
Scale Down Node	Yes (Compute nodes only)
Scale Up Nodes	Yes (Compute nodes only)
Nodes Policy Enforcement	Yes
Nodes Evacuate	Yes
OpenStack Upgrade	No

6.3. Cloud Providers

The following section outlines the status of support for Infrastructure management features on cloud providers.

6.3.1. OpenStack Platform

The following table outlines the status of support for Infrastructure management features on OpenStack Platform (OSP) cloud providers.

- The following feature support information for IBM Cloud is based on:
  - IBM Cloud PowerVC Manager 1.4.4.0 Build: 20191205-1007
  - IBM Cloud Infrastructure Center 1.2.0

Table 6.6. OpenStack Platform Feature Support

Feature	Red Hat OSP 10	Red Hat OSP 13	Red Hat OSP 16	OpenStack	IBM PowerVC	IBM Cloud Infrastructure Center
Relationship Discovery	Yes	Yes	Yes	Yes	Yes	Yes
Instance Drift Comparison	Yes	Yes	Yes	Yes	No	No
Track Instance Genealogy	Yes	Yes	Yes	Yes	Yes	Yes
Capacity & Utilization	Yes	Yes	Yes	Yes	Yes	Yes
Capture Cloud Layer Timelines	Yes	Yes	Yes	Yes	Yes	Yes
Capture Instance Event Timelines	Yes	Yes	Yes	Yes	Yes	Yes
Disk Addition to Instance	Yes	Yes	Yes	Yes	N/A	Yes
Key Pairs Inventory	Yes	Yes	Yes	Yes	Yes	Yes
Key Pairs Management	Yes	Yes	Yes	Yes	Yes	Yes
Reporting	Yes	Yes	Yes	Yes	Yes	Yes
Right Sizing	Yes	Yes	Yes	Yes	Yes	Yes
Chargeback	Yes	Yes	Yes	Yes	Yes	Yes
Remote Console Instance Access	Yes	Yes	Yes	Yes (VNC)	Yes (for NovaLink-managed hosts only)	No
Snapshot Creation and Removal	Yes	Yes	Yes	Yes	Yes	Yes
Instance Compliance Enforcement	Yes	Yes	Yes	Yes	No	No
Instance Policy Enforcement	Yes	Yes	Yes	Yes	Yes	Yes
Instance Power Operations	Yes	Yes	Yes	Yes	Yes	Yes
Instance Retirement	Yes	Yes	Yes	Yes	Yes	Yes
Alerts - Real Time	Yes	Yes	Yes	Yes	Yes	Yes
Alerts - Instance Value Changed	No	No	No	No	No	No
Alerts - Reconfigured	No	No	No	No	No	No
Integrate with Service Catalogs	Yes	Yes	Yes	Yes	Yes	Yes
Virtual Machine Reconfiguration	Yes	Yes	Yes	Yes	Yes	Yes
Volume Inventory	Yes	Yes	Yes	Yes	Yes	Yes
Volume Creation/Deletion	Yes	Yes	Yes	Yes	Yes	Yes
Instance Migration	Yes	Yes	Yes	Yes	Yes	Yes**

Feature	Red Hat OSP 10	Red Hat OSP 13	Red Hat OSP 16	OpenStack	IBM PowerVC	IBM Cloud Infrastructure Center
Automation Work Flows	Yes	Yes	Yes	Yes	Yes	Yes
Tag Mapping from Provider	No	No	No	No	Yes	No
Tag Mapping to Provider	No	No	No	No	No	No
Storage Manager	Yes	Yes	Yes	Yes	Yes	Yes
Provision Instance using PXE	No	No	No	N/A	N/A	N/A
Provision Instance using ISO	No	No	No	N/A	N/A	N/A
Provision from Template/Image to Instance	Yes	Yes	Yes	Yes	Yes	Yes
Provision from Image Snapshot to Instance	Yes	Yes	Yes	Yes	No	N/A
Provision from Volume to Instance	Yes	Yes	Yes	Yes	No	N/A
Provision from Volume Snapshot to Instance	Yes	Yes	Yes	Yes	No	N/A
Clone from Instance to Instance	No	No	No	No	No	No
Cloud-Init	Yes	Yes	Yes	Yes	Yes	N/A
Sysprep Windows Templates	Yes	Yes	Yes	Yes	N/A	N/A

Note:

Features marked with "*" are only supported with z/VM hypervisor.

Features marked with "**" are only supported with KVM hypervisor.

Features not marked with "*" or "**" are valid and supported for both z/VM and KVM hypervisors.

Features listed with a “NO” are not valid for both KVM and z/VM.

6.3.2. IBM Cloud Feature Support

The following table outlines the status of support for Infrastructure management features on IBM Cloud providers.

Table 6.7. IBM Cloud Feature Support

Feature	IBM Cloud VPC	IBM Power Systems Virtual Server
Relationship Discovery	Yes	Yes
Instance Drift Comparison	No	No
Track Instance Genealogy	Yes	Yes
Capacity & Utilization	No	No
Capture Cloud Layer Timelines	No	No
Capture Instance Event Timelines	Yes	No
Disk Addition to Instance	No	Yes
Key Pairs Inventory	Yes	Yes
Key Pairs Management	No	Yes
Reporting	Yes	Yes
Right Sizing	No	No
Chargeback	Yes (Limited to inventory allocation)	Yes (Limited to inventory allocation)
Remote Console Instance Access	No	No
Snapshot Creation and Removal	No	No
Instance Compliance Enforcement	No	No
Instance Policy Enforcement	Yes	Yes
Instance Power Operations	Yes	Yes
Instance Retirement	No	Yes
Alerts - Real Time	No	No
Alerts - Instance Value Changed	No	No
Alerts - Reconfigured	No	No
Integrate with Service Catalogs	No	Yes
Virtual Machine Reconfiguration	No	No
Volume Inventory	Yes	Yes
Volume Creation/Deletion	Yes	Yes

Feature	IBM Cloud VPC	IBM Power Systems Virtual Server
Instance Migration	N/A	N/A
Automation Work Flows	No	No
Tag Mapping from Provider	Yes	No
Tag Mapping to Provider	No	No
Storage Manager	Yes	Yes
Provision Instance using PXE	No	No
Provision Instance using ISO	No	No
Provision from Template/Image to Instance	Yes	Yes
Provision from Image Snapshot to Instance	No	No
Provision from Volume to Instance	No	No
Provision from Volume Snapshot to Instance	No	No
Clone from Instance to Instance	No	No
Cloud-Init	No	Yes
Sysprep Windows Templates	N/A	N/A

6.3.3. Amazon EC2

The following table outlines the status of support for Infrastructure management features on Amazon EC2 cloud providers.

Table 6.8. Amazon EC2 Feature Support

Feature	Amazon EC2
Relationship Discovery	Yes
VM Drift Comparison	Yes
Track VM Genealogy	Yes
Capacity & Utilization	Yes
Capture VM/Instance Event Timelines	Yes
Disk Addition to VM/Instance	Yes
Key Pairs Inventory	Yes
Key Pairs Management	Yes
Reporting	Yes
Right Sizing	Yes
Chargeback	Yes
Remote Console VM Access	No
Snapshot Creation and Removal	No
VM / Instance Compliance Enforcement	Yes
VM / Instance Policy Enforcement	Yes
VM / Instance Power Operations	Yes
VM / Instance Retirement	Yes
Alerts - Real Time	No
Alerts - VM Value Changed	No
Alerts - Reconfigured	No
Integrate with Service Catalogs	Yes
Virtual Machine Reconfiguration	No
Volume Inventory	No
Volume Creation/Deletion	No
VM Migration	No
Automation Work Flows	Yes
Tag Mapping from Provider	Yes
Tag Mapping to Provider	No
Network Manager - Read Only	Yes
Network Manager - CRUD Actions Supported	No
Storage Manager	Yes
Provision VM/Instance using PXE	No
Provision VM/Instance using ISO	No
Provision from Template/Image to VM/Instance	Yes
Clone from VM/Instance to VM/Instance	No

6.3.4. Microsoft Azure

The following table outlines the status of support for Infrastructure management features on Microsoft Azure cloud providers.

Table 6.9. Microsoft Azure Feature Support

Feature	Microsoft Azure
Relationship Discovery	Yes
VM Drift Comparison	Yes
Track VM Genealogy	Yes
Capacity & Utilization	Yes
Capture Cloud Layer Timelines	N/A
Capture VM/Instance Event Timelines	Yes
Disk Addition to VM/Instance	Yes
Key Pairs Inventory	No
Key Pairs Management	Yes
Optimization - Bottleneck Identification	No
Reporting	Yes
Right Sizing	Yes
Chargeback	Yes
Remote Console VM Access	No
Snapshot Creation and Removal	No
VM / Instance Compliance Enforcement	Yes
VM / Instance Policy Enforcement	Yes
VM / Instance Power Operations	Yes
VM / Instance Retirement	Yes
Alerts - Real Time	No
Alerts - VM Value Changed	No
Alerts - Reconfigured	No
Integrate with Service Catalogs	Yes
Virtual Machine Reconfiguration	No
Volume Inventory	No
Volume Creation/Deletion	No
VM Migration	No
Automation Work Flows	No
Tag Mapping from Provider	Yes
Tag Mapping to Provider	No
Network Manager - Read Only	Yes
Network Manager - CRUD Actions Supported	No
Provision VM/Instance using PXE	No
Provision VM/Instance using ISO	No
Provision from Template/Image to VM/Instance	Yes
Clone from VM/Instance to VM/Instance	No

6.3.5. Google Compute Engine

The following table outlines the status of support for Infrastructure Management features on Google Compute Engine cloud providers.

Table 6.10. Google Compute Engine Feature Support

Feature	Google Compute Engine
Relationship Discovery	Yes
VM Drift Comparison	No
Track VM Genealogy	Yes
Capacity & Utilization	Yes
Capture Cloud Layer Timelines	No
Capture VM/Instance Event Timelines	Yes
Disk Addition to VM/Instance	No
Key Pairs Inventory	Yes
Key Pairs Management	No
Optimization - Bottleneck Identification	No
Reporting	Yes

Feature	Google Compute Engine
Right Sizing	Yes
Chargeback	Yes
Remote Console VM Access	No
Snapshot Creation and Removal	No
VM / Instance Compliance Enforcement	Yes
VM / Instance Policy Enforcement	Yes
VM / Instance Power Operations	Yes
VM / Instance Retirement	Yes
Alerts - Real Time	No
Alerts - VM Value Changed	No
Alerts - Reconfigured	No
Integrate with Service Catalogs	Yes
Virtual Machine Reconfiguration	No
Volume Inventory	Yes
Volume Creation/Deletion	No
VM Migration	No
Automation Work Flows	Yes
Tag Mapping from Provider	No
Tag Mapping to Provider	No
Network Manager - Read Only	Yes
Network Manager - CRUD Actions Supported	No
Storage Manager	No
Provider TLS	No
Provision VM/Instance using PXE	No
Provision VM/Instance using ISO	No
Provision from Template/Image to VM/Instance	Yes
Provision from VM/Instance to Template/Image	No
Host Power Operations	No
Provision Host	No
Clone from VM/Instance to VM/Instance	No

6.3.6. Oracle Cloud

The following table outlines the status of support for Infrastructure Management features on Oracle cloud providers.

Table 6.11. Oracle Cloud Feature Support

Feature	Oracle Cloud
Relationship Discovery	Yes
VM Drift Comparison	No
Track VM Genealogy	Yes
Capacity & Utilization	Yes
Capture Cloud Layer Timelines	No
Capture VM/Instance Event Timelines	Yes
Disk Addition to VM/Instance	No
Key Pairs Inventory	Yes
Key Pairs Management	No
Optimization - Bottleneck Identification	No
Reporting	Yes
Right Sizing	No
Chargeback	No
Remote Console VM Access	No
Snapshot Creation and Removal	No
VM / Instance Compliance Enforcement	No
VM / Instance Policy Enforcement	Yes
VM / Instance Power Operations	Yes
VM / Instance Retirement	Yes
Alerts - Real Time	No

Feature	Oracle Cloud
Alerts - VM Value Changed	No
Alerts - Reconfigured	No
Integrate with Service Catalogs	Yes
Virtual Machine Reconfiguration	No
Volume Inventory	Yes
Volume Creation/Deletion	No
VM Migration	No
Automation Work Flows	Yes
Tag Mapping from Provider	No
Tag Mapping to Provider	No
Network Manager - Read Only	Yes
Network Manager - CRUD Actions Supported	No
Storage Manager	No
Provider TLS	No
Provision VM/Instance using PXE	No
Provision VM/Instance using ISO	No
Provision from Template/Image to VM/Instance	Yes
Provision from VM/Instance to Template/Image	No
Host Power Operations	No
Provision Host	No
Clone from VM/Instance to VM/Instance	No

6.4. Container Providers

The following section outlines the status of support for Infrastructure management features on Red Hat OpenShift Container Platform container providers.

Table 6.12. Red Hat OpenShift Container Platform Provider Feature Support

Feature	Red Hat OpenShift Container Platform 4.x
Relationship Discovery	Yes
Track Container Image Relationship	Yes
Capacity & Utilization	Yes
Capture Container Event Timelines	Yes
Reporting	Yes
Chargeback	Yes
Remote Cockpit Node Access	No
Container Node Compliance Enforcement	Yes
Container Node Policy Enforcement	Yes
Replicator Compliance Enforcement	Yes
Replicator Policy Enforcement	Yes
Pod Compliance Enforcement	Yes
Pod Policy Enforcement	Yes
Image Compliance Enforcement	Yes
Image Policy Enforcement	Yes
Template Provisioning	No
Smart State	No
OpenSCAP Execution and Report	No
Cross Linking Underlying Infrastructure	Yes
Tag Mapping From Provider	No
Tag Mapping To Provider	Yes

6.5. Network Providers

This section outlines the status of support for Infrastructure management features on network providers.

Table 6.13. Network Provider Feature Support

Feature	Red Hat OpenStack Platform (OSP)	Amazon EC2	Microsoft Azure	Google Compute Engine	Nuage	VMware NSX-t

Feature	Red Hat OpenStack Platform (OSP)	Amazon EC2	Microsoft Azure	Google Compute Engine	Nuage	VMware NSX-t
Network Manager Relationships	Yes	Yes	Yes	Yes	Yes	Yes
Relationship Discovery	Yes	Yes	Yes	Yes	Yes	Yes
Cloud Network Inventory	Yes	Yes	Yes	Yes	Yes	Yes
Cloud Subnet Inventory	Yes	Yes	Yes	Yes	Yes	Yes
Network Router Inventory	Yes	No	No	Yes	Yes	Yes
Security Groups	Yes	Yes	Yes	Yes	No	Yes
Floating IP Addresses	Yes	Yes	Yes	Yes	Yes	No
Network Ports	Yes	Yes	Yes	Yes	Yes	Yes
Load Balancer Inventory	No	Yes	Yes	Yes	No	No
Create/Update/Delete Network	Yes	No	No	No	Yes	Yes
Create/Update/Delete Subnet	Yes	No	No	No	Yes	No
Create/Update/Delete Routers	Yes	No	No	No	Yes	Yes

6.6. Configuration Management Providers

The following table outlines the status of support for Infrastructure management features on configuration management providers.

Table 6.14. Configuration Management Provider Feature Support

Feature	Ansible
Groups Inventory	Yes
Configured Systems Inventory	Yes
Job Template Inventory	Yes
Integrate with Service Catalog	Yes
Automate Workflows	Yes

Note: Infrastructure management supports Ansible 3.7 and later as a configuration management provider.

Chapter 7. Remote Console Support

Infrastructure management allows you to access the console of virtual machines and instances on the providers it managers using HTML5-based VNC consoles. This section outlines the status of support for remote consoles in Infrastructure management.

Note: For VMware vSphere providers, Infrastructure management supports the versions of VMRC that VMware supports. However, VMware does not support VMRC for Mac operating systems. See VMware VMRC documentation for more information.

For all other providers, Infrastructure management supports HTML5-based VNC consoles. The following have been tested with Infrastructure Management Engine:

- Windows: UltraVNC 1.1.8.8 - http://www.uvnc.com/
- Linux/Mac: SSVNC 1.0.29 - At the time of this writing, this is the only client that supports the VNC repeater configuration.

Note: As of 1 September 2015, new versions of Chrome no longer support VMRC consoles. VMRC requires NPAPI plug-in support. Chrome removed support for NPAPI plug-ins in version 45.0.2454, which was released on this date.

7.1. Support on Red Hat Enterprise Linux by Providers and Browsers

The following tables outline the status of support for remote consoles on providers for Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7.

Table 7.1. Remote Console Support on Red Hat Enterprise Linux 6

Provider	Connection Type	Support	Chrome	Firefox
VMware vSphere 6.0	VNC	Yes	No	Yes
	VMRC		No	Yes
	WebMKS		No	Yes
VMware vSphere 6.5	VMRC	Yes	No	Yes

Provider	Connection Type	Support	Chrome	Firefox
	WebMKS		No	Yes
Red Hat Virtualization (RHV) 4.2	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.3	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.4	SPICE	Yes	No	No
Red Hat OpenStack Platform 10	VNC	Yes	No	No
Red Hat OpenStack Platform 13	VNC	Yes	No	No
Red Hat OpenStack Platform 16	VNC	Yes	No	No

Table 7.2. Remote Console Support on Red Hat Enterprise Linux 7

Provider	Connection Type	Support	Chrome	Firefox
VMware vSphere 6.0	VNC	Yes	Yes	Yes
	VMRC		Yes	Yes
	WebMKS		Yes	Yes
VMware vSphere 6.5	VMRC	Yes	Yes	Yes
	WebMKS		Yes	Yes
Red Hat Virtualization (RHV) 4.2	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.3	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.4	SPICE	Yes	No	No
Red Hat OpenStack Platform 10	VNC	Yes	No	No
Red Hat OpenStack Platform 13	VNC	Yes	No	No
Red Hat OpenStack Platform 16	VNC	Yes	No	No

7.2. Support on Fedora by Providers and Browsers

The following tables outline the status of support for remote consoles on providers for Fedora 25, 26, and 27.

Table 7.3. Remote Console Support on Fedora 25

Provider	Connection Type	Support	Chrome	Firefox
VMware vSphere 6.0	VNC	Yes	Yes	Yes
	VMRC		Yes	Yes
	WebMKS		Yes	Yes
VMware vSphere 6.5	VMRC	Yes	Yes	Yes
	WebMKS		Yes	Yes
Red Hat Virtualization (RHV) 4.2	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.3	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.4	SPICE	Yes	No	No
Red Hat OpenStack Platform 10	VNC	Yes	No	No
Red Hat OpenStack Platform 13	VNC	Yes	No	No
Red Hat OpenStack Platform 16	VNC	Yes	No	No

Table 7.4. Remote Console Support on Fedora 26

Provider	Connection Type	Support	Chrome	Firefox
VMware vSphere 6.0	VNC	Yes	Yes	Yes
	VMRC		Yes	Yes
	WebMKS		Yes	Yes
VMware vSphere 6.5	VMRC	Yes	Yes	Yes
	WebMKS		Yes	Yes
Red Hat Virtualization (RHV) 4.2	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.3	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.4	SPICE	Yes	No	No
Red Hat OpenStack Platform 10	VNC	Yes	No	No
Red Hat OpenStack Platform 13	VNC	Yes	No	No
Red Hat OpenStack Platform 16	VNC	Yes	No	No

Table 7.5. Remote Console Support on Fedora 27

Provider	Connection Type	Support	Chrome	Firefox

Provider	Connection Type	Support	Chrome	Firefox
VMware vSphere 6.0	VNC	Yes	Yes	Yes
	VMRC		Yes	Yes
	WebMKS		Yes	Yes
VMware vSphere 6.5	VMRC	Yes	Yes	Yes
	WebMKS		Yes	Yes
Red Hat Virtualization (RHV) 4.2	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.3	SPICE	Yes	No	No
Red Hat Virtualization (RHV) 4.4	SPICE	Yes	No	No
Red Hat OpenStack Platform 10	VNC	Yes	No	No
Red Hat OpenStack Platform 13	VNC	Yes	No	No
Red Hat OpenStack Platform 16	VNC	Yes	No	No

7.3. Support on Windows by Providers and Browsers

The following tables outline the status of support for remote consoles on providers for Windows 7, 10, and 2012 Server.

Table 7.6. Remote Console Support on Windows 7

Provider	Connection Type	Support	Chrome	Firefox	IE 11	IE 10
VMware vSphere 6.0	VNC	Yes	Yes	Yes	Yes	No
	VMRC		Yes	Yes	Yes	No
	WebMKS		Yes	Yes	Yes	No
VMware vSphere 6.5	VMRC	Yes	Yes	Yes	Yes	No
	WebMKS		Yes	Yes	Yes	No
Red Hat Virtualization (RHV) 4.2	SPICE	Yes	No	No	No	No
Red Hat Virtualization (RHV) 4.3	SPICE	Yes	No	No	No	No
Red Hat Virtualization (RHV) 4.4	SPICE	Yes	No	No	No	No
Red Hat OpenStack Platform 10	VNC	Yes	No	No	No	No
Red Hat OpenStack Platform 13	VNC	Yes	No	No	No	No
Red Hat OpenStack Platform 16	VNC	Yes	No	No	No	No
Amazon EC2	N/A	No	N/A	N/A	N/A	N/A

Table 7.7. Remote Console Support on Windows 10

Provider	Connection Type	Support	Chrome	Firefox	IE 11	IE 10	Edge
VMware vSphere 6.0	VNC	Yes	No	No	No	No	Yes
	VMRC		No	No	No	No	Yes
	WebMKS		No	No	No	No	Yes
VMware vSphere 6.5	VMRC	Yes	No	No	No	No	Yes
	WebMKS		No	No	No	No	Yes
Red Hat Virtualization (RHV) 4.2	SPICE	Yes	No	No	No	No	No
Red Hat Virtualization (RHV) 4.3	SPICE	Yes	No	No	No	No	No
Red Hat Virtualization (RHV) 4.4	SPICE	Yes	No	No	No	No	No
Red Hat OpenStack Platform 10	VNC	Yes	No	No	No	No	No
Red Hat OpenStack Platform 13	VNC	Yes	No	No	No	No	No
Red Hat OpenStack Platform 16	VNC	Yes	No	No	No	No	No

Provider	Connection Type	Support	Chrome	Firefox	IE 11	IE 10	Edge
Amazon EC2	N/A	No	N/A	N/A	N/A	N/A	N/A

Table 7.8. Remote Console Support on Windows 2012 Server

Provider	Connection Type	Support	Chrome	Firefox	IE 11	IE 10
VMware vSphere 6.0	VNC	Yes	Yes	Yes	Yes	No
	VMRC		No	No	No	No
	WebMKS		No	No	No	No
VMware vSphere 6.5	VMRC	Yes	Yes	Yes	Yes	No
	WebMKS		Yes	Yes	Yes	No
Red Hat Virtualization (RHV) 4.2	SPICE	Yes	No	No	No	No
Red Hat Virtualization (RHV) 4.3	SPICE	Yes	No	No	No	No
Red Hat Virtualization (RHV) 4.4	SPICE	Yes	No	No	No	No
Red Hat OpenStack Platform 10	VNC	Yes	No	No	No	No
Red Hat OpenStack Platform 13	VNC	Yes	No	No	No	No
Red Hat OpenStack Platform 16	VNC	Yes	No	No	No	No
Amazon EC2	N/A	No	N/A	N/A	N/A	N/A

Chapter 8. Operating System Support

This section outlines the supported operating systems by infrastructure and cloud providers.

Table 8.1. Windows Support by Infrastructure Providers

Infrastructure Provider	Windows 7 SP1	Windows 8.1	Windows 10	Windows 2012	Windows 2016
Red Hat Virtualization 4.2			Yes		
Red Hat Virtualization 4.3			Yes		
Red Hat Virtualization 4.4			Yes		
Red Hat OpenStack Platform 10 [a]	Yes	Yes	Yes	Yes	Yes
Red Hat OpenStack Platform 13 [a]	Yes	Yes	Yes	Yes	Yes
Red Hat OpenStack Platform 16 [a]	Yes	Yes	Yes	Yes	Yes
VMware vCenter 6.0 [b]	Yes	Yes	Yes	Yes	Yes
VMware vCenter 6.5 [c]	Yes	Yes	Yes	Yes	Yes
VMware vCenter 6.7 [d]	Yes	Yes	Yes	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

[b] Install the VMware VDDK 6.0 so that SmartState analysis is successful.

[c] Install the VMware VDDK 6.5 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

[d] Install the VMware VDDK 6.7 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

Table 8.2. Windows Support by Cloud Providers

Cloud Provider	Windows 7 SP1	Windows 8.1	Windows 10	Windows 2012	Windows 2016
Red Hat OpenStack Platform 10 [a]	Yes	Yes	Yes	Yes	Yes
Red Hat OpenStack Platform 13 [a]	Yes	Yes	Yes	Yes	Yes
Red Hat OpenStack Platform 16 [a]	Yes	Yes	Yes	Yes	Yes
Microsoft Azure	Yes	Yes	Yes	Yes	Yes
Amazon EC2	Yes	Yes	Yes	Yes	Yes
[a] See Section 9.2, Supported Guest Operating Systems.					

Table 8.3. Linux Support by Infrastructure Providers

Infrastructure Provider	RHEL 6	RHEL 7	Ubuntu	SUSE
Red Hat Virtualization 4.2		Yes		
Red Hat Virtualization 4.3		Yes		
Red Hat Virtualization 4.4		Yes		
Red Hat OpenStack Platform 10 [a]	Yes	Yes	Yes	Yes
Red Hat OpenStack Platform 13 [a]	Yes	Yes	Yes	Yes

Infrastructure Provider	RHEL 6	RHEL 7	Ubuntu	SUSE
Red Hat OpenStack Platform 16 [a]	Yes	Yes	Yes	Yes
VMware vCenter 6.0 [b]	Yes	Yes	Yes	Yes
VMware vCenter 6.5 [c]	Yes	Yes	Yes	Yes
VMware vCenter 6.7 [d]	Yes	Yes	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

[b] Install the VMware VDDK 6.0 so that SmartState analysis is successful.

[c] Install the VMware VDDK 6.5 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

[d] Install the VMware VDDK 6.7 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

Table 8.4. Linux Support by Cloud Providers

Cloud Provider	RHEL 6	RHEL 7	Ubuntu	SUSE
Red Hat OpenStack Platform 10 [a]	Yes	Yes	Yes	Yes
Red Hat OpenStack Platform 13 [a]	Yes	Yes	Yes	Yes
Red Hat OpenStack Platform 16 [a]	Yes	Yes	Yes	Yes
Microsoft Azure	Yes	Yes	Yes	Yes
Amazon EC2	Yes	Yes	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

Chapter 9. Smart State Analysis Support

Infrastructure management allows you to analyze virtual machines and instances on the providers it manages to collect information about the base operating system on those virtual machines or instances, their virtual hardware, installed applications, and other details. This operation is known as SmartState analysis. This section outlines the status of support for SmartState analysis of virtual machines and instances by guest operating system, file system, and by provider.

9.1. File Systems

The following tables outline SmartState analysis support for different file systems by infrastructure and cloud providers.

Table 9.1. Windows File System SmartState Analysis Support by Infrastructure Providers

Infrastructure Provider	ReFS	NTFS	FAT32	FAT
Red Hat Virtualization 4.2	No			
Red Hat Virtualization 4.3	No			
Red Hat Virtualization 4.4	No			
Red Hat OpenStack Platform 10 [a]	No	Yes	No	No
Red Hat OpenStack Platform 13 [a]	No	Yes	No	No
Red Hat OpenStack Platform 16 [a]	No	Yes	No	No
Red Hat OpenShift	No			
VMware vCenter 6.0 [b]	No	Yes	Yes	Yes
VMware vCenter 6.5 [c]	No	Yes	Yes	Yes
VMware vCenter 6.7 [d]	No	Yes	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

[b] Install the VMware VDDK 6.0 so that SmartState analysis is successful.

[c] Install the VMware VDDK 6.5 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

[d] Install the VMware VDDK 6.7 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

Table 9.2. Windows File System SmartState Analysis Support by Cloud Providers

Cloud Provider	ReFS	NTFS	FAT32	FAT
Red Hat OpenStack Platform 10 [a]	No	Yes	No	No
Red Hat OpenStack Platform 13 [a]	No	Yes	No	No
Red Hat OpenStack Platform 16 [a]	No	Yes	No	No
Microsoft Azure	No	Yes	Yes	Yes
Amazon EC2	No	Yes	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

Table 9.3. Linux File System SmartState Analysis Support by Infrastructure Providers

Infrastructure Provider	EXT3	EXT4	XFS
Red Hat Virtualization 4.2			
Red Hat Virtualization 4.3			
Red Hat Virtualization 4.4			
Red Hat OpenStack Platform 10 [a]	Yes	Yes	Yes
Red Hat OpenStack Platform 13 [a]	Yes	Yes	Yes
Red Hat OpenStack Platform 16 [a]	Yes	Yes	Yes
Red Hat OpenShift			
VMware vCenter 6.0 [b]	Yes	Yes	Yes
VMware vCenter 6.5 [c]	Yes	Yes	Yes
VMware vCenter 6.7 [d]	Yes	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

[b] Install the VMware VDDK 6.0 so that SmartState analysis is successful.

[c] Install the VMware VDDK 6.5 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

[d] Install the VMware VDDK 6.7 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

Table 9.4. Linux File System SmartState Analysis Support by Cloud Providers

Cloud Provider	EXT3	EXT4	XFS
Red Hat OpenStack Platform 10 [a]	Yes	Yes	Yes
Red Hat OpenStack Platform 13 [a]	Yes	Yes	Yes
Red Hat OpenStack Platform 16 [a]	Yes	Yes	Yes
Microsoft Azure	Yes	Yes	Yes
Amazon EC2	Yes	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

Table 9.5. Other File System SmartState Analysis Support by Infrastructure Providers

Infrastructure Provider	CDFS	Any Encrypted FS
Red Hat Virtualization 4.2		
Red Hat Virtualization 4.3		
Red Hat Virtualization 4.4		
Red Hat OpenStack Platform 10 [a]	Yes	Yes
Red Hat OpenStack Platform 13 [a]	Yes	Yes
Red Hat OpenStack Platform 16 [a]	Yes	Yes
Red Hat OpenShift		
VMware vCenter 6.0 [b]	Yes	Yes
VMware vCenter 6.5 [c]	Yes	Yes
VMware vCenter 6.7 [d]	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

[b] Install the VMware VDDK 6.0 so that SmartState analysis is successful.

[c] Install the VMware VDDK 6.5 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

[d] Install the VMware VDDK 6.7 so that SmartState analysis is successful. Alternatively, you can use VMware VDDK 6.0.

Table 9.6. Other File System SmartState Analysis Support by Cloud Providers

Cloud Provider	CDFS	Any Encrypted FS
Red Hat OpenStack Platform 10 [a]	Yes	Yes
Red Hat OpenStack Platform 13 [a]	Yes	Yes
Red Hat OpenStack Platform 16 [a]	Yes	Yes
Microsoft Azure	Yes	Yes
Amazon EC2	Yes	Yes

[a] See Section 9.2, Supported Guest Operating Systems.

9.2. Guest Operating Systems

Table 9.7. Supported Guest Operating Systems With Their Support Life Cycle

Guest Operating System	EOL Date	Extended Support Date	Link
Windows 2012 R2 Standard	10/9/2018	10/10/2023	Windows 2012 R2
Windows 2012 R2 Datacenter	10/9/2018	10/10/2023	Windows 2012 R2
Windows 2016 Standard	1/11/2022	1/11/2027	Windows 2016 Standard
Windows 2016 Datacenter	1/11/2022	1/11/2027	Windows 2016 Datacenter
Windows 7 SP1	1/13/2015	1/14/2020	Windows 7 SP1
Windows 8.1	1/9/2018	1/10/2023	Windows 8.1
Windows 10 Ent. 2016 LTSB	10/12/2021	10/13/2026	Windows 10 Ent. 2016
Red Hat OpenStack Platform 10.0	6/16/2018	12/16/2019	RHOSP Life Cycle
Red Hat OpenStack Platform 13.0	12/27/2019	6/27/2021	RHOSP Life Cycle
Red Hat OpenStack Platform 16.0	10/27/2020	4/30/2024	RHOSP Life Cycle

- Deployment Planning Guide
  - Introduction to Infrastructure Automation
    - Architecture
    - Requirements
      - Virtual Hardware Requirements
      - Database Requirements
      - Browser Requirements
      - Additional Requirements
    - Terminology
    - Regions
    - Roles
      - Appliance Types
    - Centralized Administration
    - Tenancy
    - Using a Load Balancer
      - Configuring the Load Balancer
      - Configuring Worker Appliances for Load Balancing
    - Database Configuration
      - User Overrides
      - Reading Configuration Settings
  - Capacity Planning
    - Capacity and Utilization Collection
    - Assigning the Capacity and Utilization Server Roles
    - Adding Database Credentials for Data Collection
    - Data Collection for Red Hat Virtualization
    - Data Collection for Red Hat Enterprise Linux OpenStack Platform
    - Capacity and Utilization Data Collected
      - Capacity and Utilization Charts for Hosts, Clusters, and Virtual Machines
      - Capacity and Utilization Charts for Datastores
    - Capacity and Utilization Chart Features
      - Zooming into a Chart
      - Drilling into Chart Data
    - Planning Where to Put a New Virtual Machine
- Appendix
  - Example PostgreSQL Configuration File

Deployment Planning Guide

Introduction to Infrastructure Automation

Infrastructure Automation delivers the insight, control, and automation that enterprises need to address the challenges of managing virtual environments. This technology enables enterprises with existing virtual infrastructures to improve visibility and control, and those starting virtualization deployments to build and operate a well-managed virtual infrastructure.

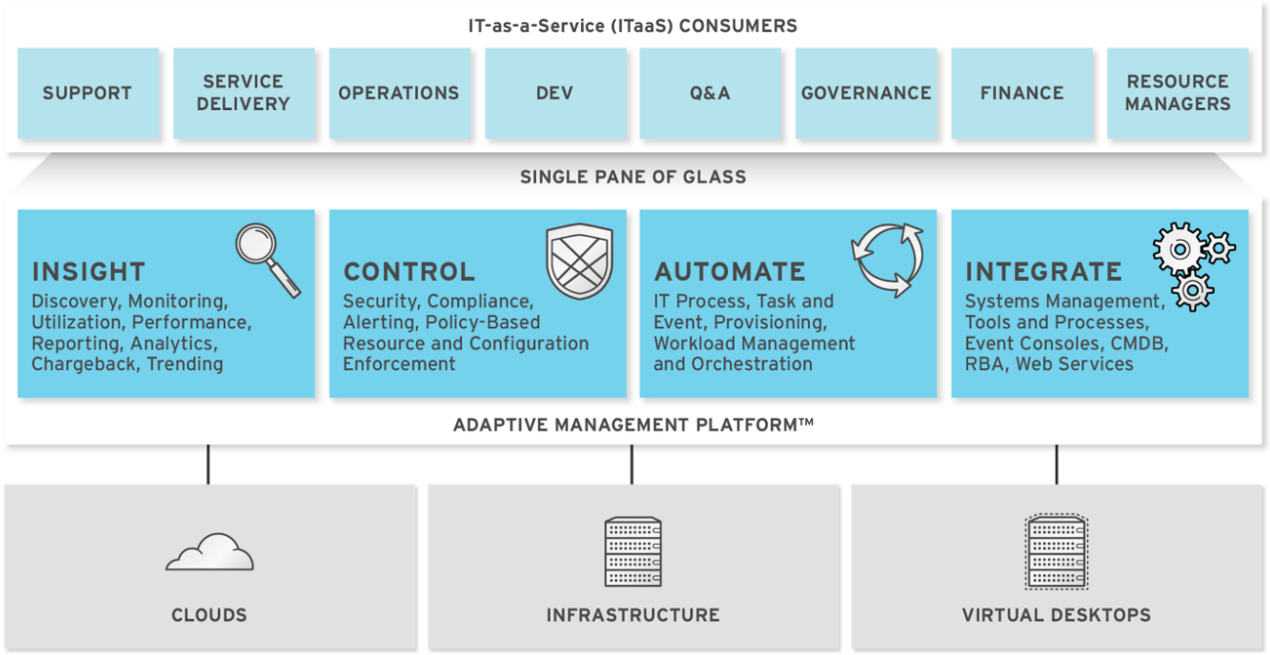
Infrastructure Automation provides the following feature sets:

- Insight: Discovery, Monitoring, Utilization, Performance, Reporting, Analytic, Chargeback, and Trending.

- Control: Security, Compliance, Alerting, and Policy-Based Resource, and Configuration Enforcement.
- Automate: IT Process, Task and Event, Provisioning, and Workload Management and Orchestration.
- Integrate: Systems Management, Tools and Processes, Event Consoles, Configuration Management Database (CMDB), Role-based Administration (RBA), and Web Services.

Architecture

The diagram below describes the capabilities of Infrastructure Automation. Its features are designed to work together to provide robust management and maintenance of your virtual infrastructure.



The architecture comprises the following components:

- The Infrastructure Automation appliance (appliance) which is supplied as a secure, high-performance, preconfigured virtual machine. It provides support for HTTPS communications.
- The Infrastructure Automation Server (Server) resides on the appliance. It is the software layer that communicates between the SmartProxy and the Virtual Management Database. It includes support for HTTPS communications.
- The Virtual Management Database (VMDB) resides either on the appliance or another computer accessible to the appliance. It is the definitive source of intelligence collected about your Virtual Infrastructure. It also holds status information regarding appliance tasks.
- The Infrastructure Automation Console (Console) is the Web interface used to view and control the Server and appliance. It is consumed through Web 2.0 mash-ups and web services (WS Management) interfaces.
- The SmartProxy can reside on the appliance or on an ESX Server. If not embedded in the Server, the SmartProxy can be deployed from the appliance. A SmartProxy agent must configured in each storage location, and must be visible to the appliance. The SmartProxy acts on behalf of the appliance communicating with it over HTTPS on standard port 443.

Requirements

To use Infrastructure Automation, certain virtual hardware, database, and browser requirements must be met in your environment.

Virtual Hardware Requirements

The Infrastructure Automation appliance requires the following virtual hardware at minimum:

- 4 VCPUs
- 12 GB RAM
- 44 GB HDD + optional database disk

Database Requirements

Red Hat recommends allocating the virtual machine disk fully at the time of creation. Three main factors affect the size of your database over time:

- Virtual Machine Count: the most important factor in the calculation of virtual machine database (VMDB) size over time.
- Host Count: the number of hosts associated with the provider.
- Storage Count: the number of individual storage elements as seen from the perspective of the provider or host. It is not the total number of virtual disks for all virtual machines.

Use the following table as a guideline to calculate minimum requirements for your database:

			Estimated VMDB Size in GB	
Virtual Machine Count	Host Count	Storage Count	1 year	2 year
100	5	50	3.5	5
500	10	100	17	25
5000	50	500	173	251

Note:

When enabling capacity and utilization for metrics gathering over a period of time, it is recommended that the VMDB size scale accordingly. Evaluate the number of instances in your provider inventory and storage duration requirements to plan for increased VMDB sizing requirements.

Use the following information to plan for your increased VMDB needs when working with metrics gathering:

- Realtime metrics data are stored for 4 hours.
- Rollup metrics data are stored for 6 months.

Example:

	Minute	**Hour**	**Day**
OpenStack Provider Instance	**3** Realtime Metrics	**181** (3 records * 60 minutes = 180 Realtime Metrics + 1 hourly Rollup Metric)	**4,345** (3 records * 60 minutes * 24 hours =4320 Realtime Metrics + 1 daily Rollup Metric)

- Metrics data storage times can be configured by editing the Advanced Settings.

Browser Requirements

To use Infrastructure Automation, the following browser requirements must be met:

- One of the following web browsers:
  - Google Chrome
  - Mozilla Firefox
  - Safari
  - Internet Explorer 10 or higher

Additional Requirements

Additionally, the following must be configured to use Infrastructure Automation:

- The Infrastructure Automation appliance must already be installed and activated in your enterprise environment.
- The SmartProxy must have visibility into the virtual machines and cloud instances that you want to control.
- For more information, see SmartProxies in the Infrastructure Automation *General Configuration* guide.

Terminology

The following terms are used throughout the documentation. Review them before proceeding.

- Account Role The level of access a user has to different parts and functions of the Infrastructure Automation console. There are a variety of Account Roles, which can be assigned to users to restrict or allow access to parts of the console and virtual infrastructure.
- Action An execution that is performed after a condition is evaluated.

- Alert Infrastructure Automation alerts notify administrators and monitoring systems of critical configuration changes and threshold limits in the virtual environment. The notification can take the form of either an email or an SNMP trap.
- Analysis Profile A customized scan of hosts, virtual machines, or instances. You can collect information from categories, files, event logs, and registry entries.
- Cloud A pool of on-demand and highly available computing resources. The usage of these resources are scaled depending on the user requirements and metered for cost.
- Infrastructure Automation Appliance A virtual machine where the virtual management database (VMDB) and Infrastructure Automation reside.
- Infrastructure Automation Console A web-based interface into the Infrastructure Automation appliance.
- Infrastructure Automation Role A designation assigned to a Infrastructure Automation server that defines what a Infrastructure Automation server can do.
- Infrastructure Automation Server The application that runs on the Infrastructure Automation appliance and communicates with the SmartProxy and the VMDB.
- Cluster Hosts that are grouped together to provide high availability and load balancing.
- Condition A control policy test triggered by an event, which determines a subsequent action.
- Discovery Process run by the Infrastructure Automation server which finds virtual machine and cloud providers.
- Drift The comparison of a virtual machine, instance, host, cluster to itself at different points in time.
- Event A trigger to check a condition.
- Event Monitor Software on the Infrastructure Automation appliance which monitors external providers for events and sends them to the Infrastructure Automation server.
- Host A computer running a hypervisor, capable of hosting and monitoring virtual machines. Supported hypervisors include RHV-H, VMware ESX hosts, Windows Hyper-V hosts.
- Instance/Cloud Instance A on-demand virtual machine based upon a predefined image and uses a scalable set of hardware resources such as CPU, memory, networking interfaces.
- Managed/Registered VM A virtual machine that is connected to a host and exists in the VMDB. Also, a template that is connected to a provider and exists in the VMDB. Note that templates cannot be connected to a host.
- Managed/Unregistered VM A virtual machine or template that resides on a repository or is no longer connected to a provider or host and exists in the VMDB. A virtual machine that was previously considered registered may become unregistered if the virtual machine was removed from provider inventory.
- Provider An external management system that Infrastructure Automation integrates in order to collect data and perform operations.
- Policy A combination of an event, a condition, and an action used to manage a virtual machine.
- Policy Profile A set of policies.
- Refresh A process run by the Infrastructure Automation server which checks for relationships of the provider or host to other resources, such as storage locations, repositories, virtual machines, or instances. It also checks the power states of those resources.
- Regions A region is the collection of zones that share the same database for reporting and charting. A master region may be added to synchronize multiple VMDBs into one VMDB for higher-level reporting, providing a “single pane of glass” view.
- Resource A host, provider, instance, virtual machine, repository, or datastore.
- Resource Pool A group of virtual machines across which CPU and memory resources are allocated.
- Repository A place on a datastore resource which contains virtual machines.
- SmartProxy The SmartProxy is a software agent that acts on behalf of the Infrastructure Automation appliance to perform actions on hosts, providers, storage and virtual machines.

- The SmartProxy can be configured to reside on the Infrastructure Automation appliance or on an ESX server version. The SmartProxy can be deployed from the Infrastructure Automation appliance, and provides visibility to the VMFS storage. Each storage location must have a SmartProxy with visibility to it. The SmartProxy acts on behalf of the Infrastructure Automation appliance. If the SmartProxy is not embedded in the Infrastructure Automation server, it communicates with the Infrastructure Automation appliance over HTTPS on standard port 443.

- SmartState Analysis Process run by the SmartProxy which collects the details of a virtual machine or instance. Such details include accounts, drivers, network information, hardware, and security patches. This process is also run by the Infrastructure Automation server on hosts and clusters. The data is stored in the VMDB.

- SmartTags Descriptors that allow you to create a customized, searchable index for the resources in your clouds and infrastructure.
- Storage Location A device, such as a VMware datastore, where digital information resides that is connected to a resource.
- Tags Descriptive terms defined by a Infrastructure Automation user or the system used to categorize a resource.
- Template A template is a copy of a preconfigured virtual machine, designed to capture installed software and software configurations, as well as the hardware configuration, of the original virtual machine.
- Unmanaged Virtual Machine Files discovered on a datastore that do not have a virtual machine associated with them in the VMDB. These files may be registered to a provider that the Infrastructure Automation server does not have configuration information on. Possible causes may be that the provider has not been discovered or that the provider has been discovered, but no security credentials have been provided.
- Virtual Machine A software implementation of a system that functions similar to a physical machine. Virtual machines utilize the hardware infrastructure of a physical host, or a set of physical hosts, to provide a scalable and on-demand method of system provisioning.
- Virtual Management Database (VMDB) Database used by the Infrastructure Automation appliance to store information about your resources, users, and anything else required to manage your virtual enterprise.
- Virtual Thumbnail An image in the web interface representing a resource, such as a provider or a virtual machine, showing the resource's properties at a glance. Each virtual thumbnail is divided into quadrants, which provide information about the resource, such as its software and power state.
- Worker Appliance A Infrastructure Automation appliance dedicated to a role other than user interface or database.
- Zones Infrastructure Automation Infrastructure can be organized into zones to configure failover and to isolate traffic. Zones can be created based on your environment. Zones can be based on geographic location, network location, or function. When first started, new servers are put into the default zone. ## Planning

This guide provides some general guidelines to planning a deployment on Infrastructure Automation. This includes creating multiple regions containing Infrastructure Automation appliances, CPU sizing recommendations, database sizing recommendations, and database configuration.

Regions

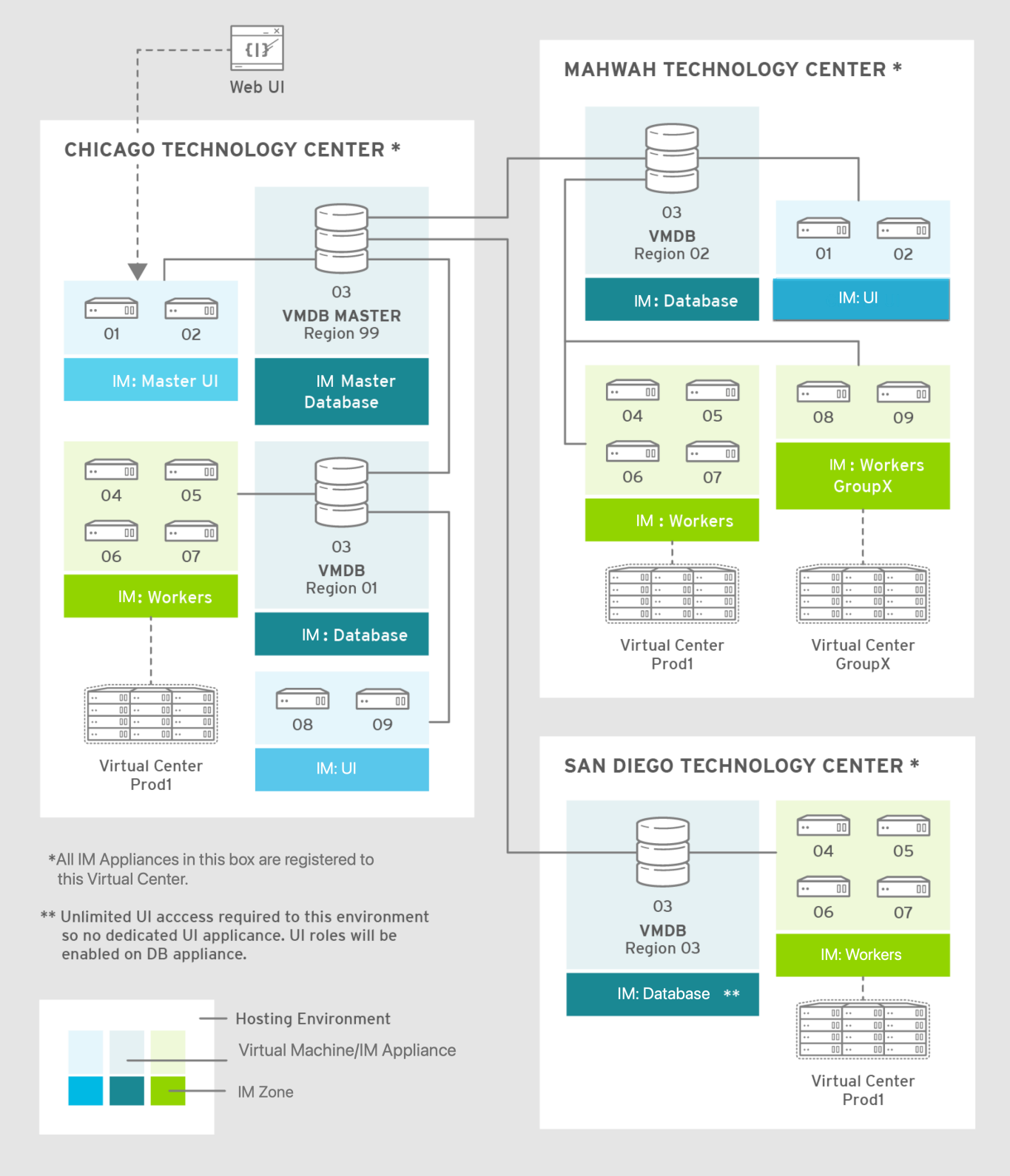
Regions are used for centralizing data which is collected from public and private virtualization environments. A region is ultimately represented as a single database for the VMDB. Regions are particularly useful when multiple geographical locations need to be managed, as they enable all the data collection to happen at each particular location and avoid data collection traffic across slow links between networks.

When multiple regions are being used, each with their own unique ID, a master region can be created to centralize the data of all the children regions into a single master database. To do this, configure each child region to replicate its data to the master region database (the recommended region is 99, though any number up to three digits will work). This parent and child region is a one-to-many relationship.

Regions can contain multiple zones, which in turn contain appliances. Zones are used for further segregating network traffic along with enabling failover configurations. Each appliance has the capability to be configured for a number of specialized server roles. These roles are limited to the zone containing the appliance they run on.

Only one failover type of each server role can run in a zone. If multiple appliances have the same failover role, the extras are used as backups that activate only if the primary appliance fails. Non-failover server roles can run on multiple appliances simultaneously in a zone, so resources can be adjusted according to the workload those roles are responsible for.

The following diagram demonstrates an example of the multiple regions working together in a Infrastructure Automation environment.

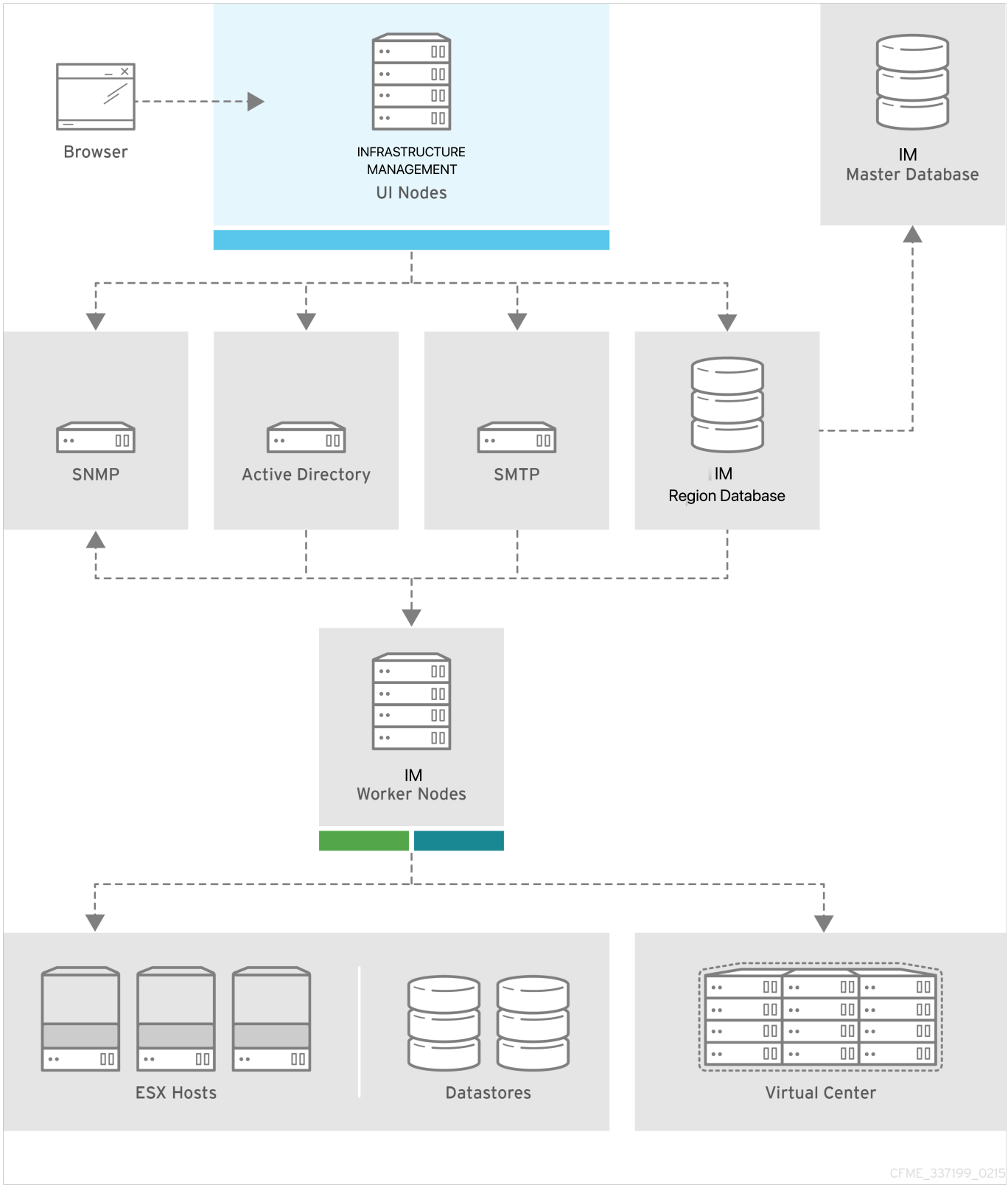


The master appliance is located in Chicago and contains a master region and a subregion that manages the worker appliances. The Mahwah technology center contains a single subregion that manages two zones. Likewise, the San Diego technology center contains a single subregion managing a single zone.

Note:

- Replicating a parent region to a higher-level parent is not supported.
- Parent regions can be configured after the child regions are online.

The following diagram provides a closer look at a region:



In this region, we have several Infrastructure Automation appliances acting as UI nodes and worker nodes. These worker nodes execute tasks on the providers in your environment. The region also uses a region database that reports to a master database on the main Infrastructure Automation appliance. All appliances can connect to the authentication services (Active Directory, LDAP, Identity Management), outgoing mail (SMTP), and network services (SNMP).

Note:

Infrastructure Automation can be configured in a highly available setup. In this case, all PostgreSQL instances must be running on a server that is deployed from the Infrastructure Automation appliance. High availability is achieved by database replication between two or more database servers.

Roles

Server roles define what a server can do. Assigning different server roles to appliances can allow them to focus on specific functions. When planning a deployment, consider which roles to assign to each appliance. Some server roles are enabled by default in Infrastructure Automation. Many server roles start worker processes.

Some roles are also dependent on other roles. For example, because the Infrastructure Automation user interface relies on the API for access, the Web Services role must be enabled with the User Interface role for users to log in to the appliance. See Server Roles in *General Configuration* for details on each server role and its function.

Appliance Types

Depending on the needs of your environment, you may choose to separate worker and database tasks between appliances. One example of this is to implement a highly available configuration so that certain appliances are running the PostgreSQL database and providing failover. For more details about configuring high availability, see the High Availability Guide.

The following provides a summary of types of appliances:

Appliance Type	Database	Workers	Description
VMDB appliance	Yes	Yes	Worker processes are running, and it also hosts its own database that other appliances can connect to.
Non-database appliance	No	Yes	Worker processes are running on the appliance, but it does not host a database. The appliance is connected to an external database.
Dedicated-database appliance	Yes	No	This appliance contains no worker processes, only a database for other appliances to connect to.
Non-Infrastructure Automation VM with database	Yes	No	This appliance contains no worker processes, only a database. As this is not a Infrastructure Automation appliance, you cannot run any Infrastructure Automation `rake` tasks on it. This appliance must be migrated using a non-database appliance that is pointed at it, using it as a database.

Appliance types

Centralized Administration

Infrastructure Automation includes centralized administration capabilities, where certain operations can be initiated from the global region and processed and executed on remote regions. From the global region, you can also access the user interfaces of virtual machines residing in remote regions.

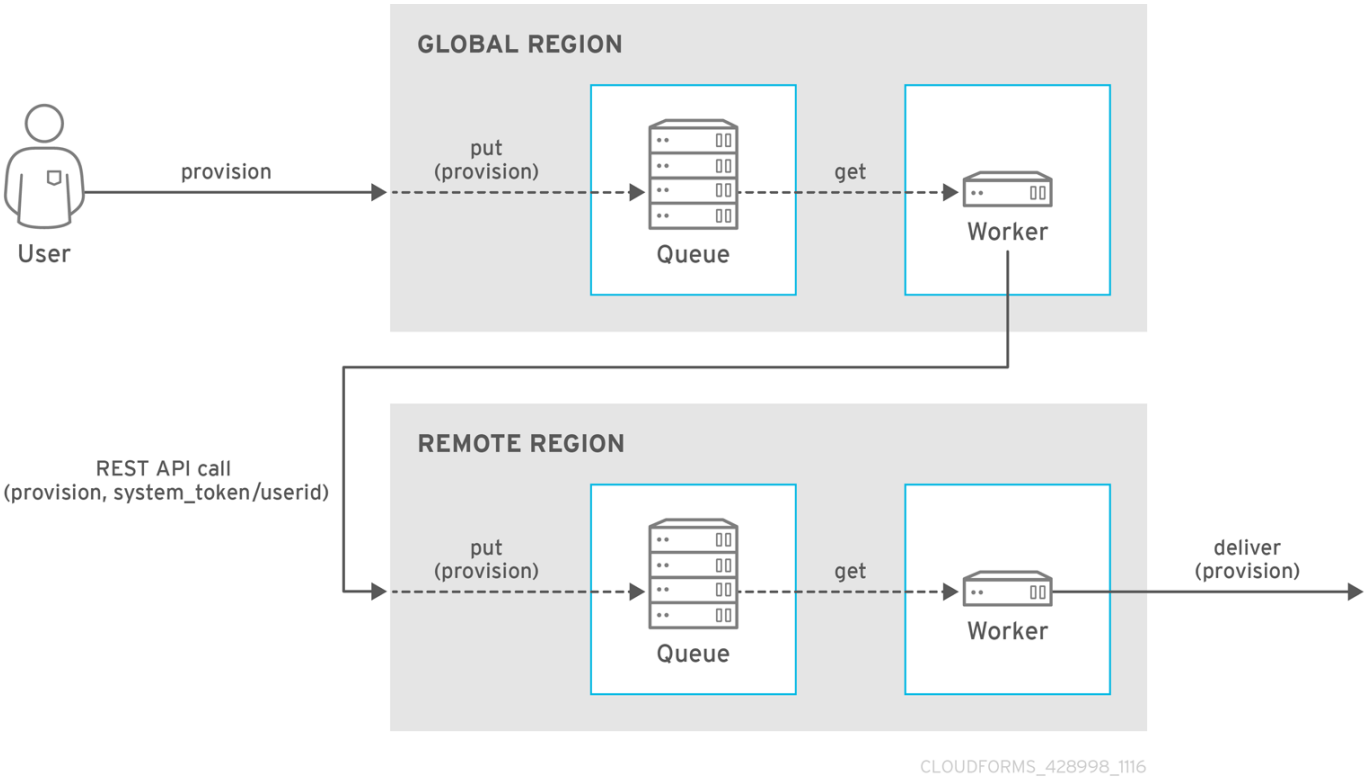
The following life cycle management operations can be initiated from the global region using centralized administration:

- Virtual machine provisioning
- Virtual machine power operations
- Virtual machine retirement
- Virtual machine reconfiguration
- Service provisioning
- Service retirement
- Opening a virtual machine in the remote region

Note:

Infrastructure Automation life cycle operations other than those listed above are not supported. Centralized administration capabilities are not supported from the Self Service user interface.

With centralized administration, the remote_queue_put leverages a new system-to-system REST API request to forward the original request to the remote region. This request is put in the local queue in the remote region, which is then delivered by a worker in the remote region as if it was queued there all along. As a result, a Infrastructure Automation operator in the global region can be seen as provisioning on behalf of a remote region.



Note:

The operations initiated from the global region are subject to the role-based access control (RBAC) rules on the remote region. The user in the remote region which matches the logged-in user's **user ID** will be used to enforce RBAC in the target region. The operation will fail on the remote system if the user does not have the required permissions.

Tenancy

Infrastructure Automation supports multitenancy. Tenants can be totally separate or they can be in a parent-child or peer relationship. Tenants in a relationship can share or inherit a certain configuration. You can subdivide and create child tenants and they, in turn, can have child tenants, and so on. The ability to have multi-level (nested) tenants in a hierarchy enables those at the bottom to inherit permissions from those above. This configuration allows for granular user permissions to be set on specific tenants.

A tenant can also contain a self-contained child tenant known as a 'project'. A project cannot have a child tenant, but is useful for allocating resources to a small group or team within a larger organization.

Note:

If you do not add any additional tenants, all resources and user accounts are contained in a single base tenant which is your Infrastructure Automation appliance itself. In Infrastructure Automation, is sometimes referred as 'tenant zero'.

Tenancy Account Roles.

In Infrastructure Automation, the following two account roles are associated with tenancy:

- Tenant administrator
- Tenant quota administrator

Tenant administrator and tenant quota administrator roles are like administrator and super administrator. These roles are not limited to the tenant upon which they are acting and act across all tenants, and therefore should be considered privileged users. These are not roles inside a tenant.
Tenancy Models.

The following two approaches exist for tenancy planning:

- **Tenantless** - You can create a single large tenant, sometimes referred as 'tenant zero', and perform all your operations in there without any subdivision of resources or user accounts.
- **Enterprise model** - A common scenario is to create a single tenant, and then subdivide it based on the structures or departments within your organization. Those departments are then able to further subdivide their resources into distinct projects. With this model, you have a single URL for user access, while still having the ability to divide resources into nested hierarchical tenants.

Tenancy Configuration.

You can create and configure tenancy using the Infrastructure Automation user interface in the same place you set up users, groups and roles by selecting **Configuration** from the settings menu, and then clicking on the **Access Control** accordion.

Tenancy in Automation.

One of the features of tenancy is that each tenant can have its own automate domain. Tenant-based domains can help in several use cases, such as if you have:

- groups that need their own naming routines
- varying types of approval needs
- departments that use different end ticketing systems
- a customer who is a holding company or centralized IT organization for managing different business units

Just like standard domains are nested, you can also add automate domains that are nested at the tenant level.

Tenancy Quota and Reporting.

You can allocate and enforce quotas for the following attributes:

- Virtual CPUs
- Memory in GB
- Storage in GB
- Number of virtual machines
- Number of templates

You can generate or schedule a report for **Tenant Quotas** similar to other reports.

Note:

Currently, in tenant quota reports you will see all of the tenants but there is no nesting information available by parent and child tenants.

Example:

In the following example of a tenant quota report, *DevOps Teams* is a parent tenant and *Team Alpha* and *Team Bravo* are child tenants.

Tenant Name	Quota Name	Total Quota	In Use	Allocated	Available
DevOps Teams	Allocated Virtual CPUs	125 Count	0 Count	75 Count	50 Count
DevOps Teams	Allocated Memory in GB	255.0 GB	0.0 GB	125.0 GB	130.0 GB
DevOps Teams	Allocated Storage in GB	1505.0 GB	0.0 GB	1100.0 GB	405.0 GB
DevOps Teams	Allocated Number of Templates	15 Count	0 Count	4 Count	11 Count
DevOps Teams	Allocated Number of Virtual Machines	160 Count	0 Count	70 Count	90 Count
DevOps Teams					
Team Alpha	Allocated Virtual CPUs	50 Count	0 Count	0 Count	50 Count
Team Alpha	Allocated Memory in GB	75.0 GB	0.0 GB	0.0 GB	75.0 GB
Team Alpha	Allocated Storage in GB	1000.0 GB	0.0 GB	0.0 GB	1000.0 GB
Team Alpha	Allocated Number of Virtual Machines	50 Count	0 Count	0 Count	50 Count
Team Alpha	Allocated Number of Templates	3 Count	0 Count	0 Count	3 Count
Team Alpha					
Team Bravo	Allocated Virtual CPUs	25 Count	0 Count	0 Count	25 Count
Team Bravo	Allocated Memory in GB	50.0 GB	0.0 GB	0.0 GB	50.0 GB
Team Bravo	Allocated Storage in GB	100.0 GB	0.0 GB	0.0 GB	100.0 GB
Team Bravo	Allocated Number of Virtual Machines	20 Count	0 Count	0 Count	20 Count
Team Bravo	Allocated Number of Templates	1 Count	0 Count	0 Count	1 Count
Team Bravo					

- Total Quota: Total quota enforced per attribute for a tenant
- In Use: Amount of quota currently in use by tenants

- Allocated: Amount of quota given to all child tenants
- Available: *Total Quota* minus (-) *In Use* minus (-) *Allocated*

Tenancy Chargeback.

You have the ability to do tenancy in chargeback where you are able to assign rates and have a different rate for each tenant. You can make use of the default rate or create your own set of rates depending on the tenant. As well, there is an ability to create chargeback reports by tenant.

Tenancy Service Catalogs.

Similar to automate domains, you can have service catalogs at each level of tenancy. Once you add a service catalog at a particular level of tenancy, it is visible to that tenant and its children (unless you use tagging to exclude).

Tenancy Providers.

Providers can be added at any level of tenancy. Once added, a provider is visible to any child or lower tenants, making it possible to easily separate resources that are owned or accessed by one group, and should not be available to other tenants.

Tenancy Relationships and Properties.

The tenant summary page found in **Configuration > Access Control > Tenants > Tenant** provides detailed information about the tenant and its relationships including:

- Catalog items and bundles
- Automate domains
- Provider relationships

Using a Load Balancer

Deploying multiple user interface worker appliances and placing them behind a third-party load balancer allows for redundancy and improved performance. This requires extra configuration in both the load balancer and in the Infrastructure Automation user interface worker appliances.

Configuring the Load Balancer

- Configure the load balancer to use sticky sessions. This ensures that when a session is started, all requests for that session are sent to the same worker appliance.
- Configure the load balancer to test for connectivity using the Infrastructure Automation ping response page: **`https://appliance_name/ping`**. The expected reply from the appliance is the text string *pong*. Using this URL is preferable to the appliance login URL as it does not establish a connection to the database.

Configuring Worker Appliances for Load Balancing

When using a load balancer, configure appliances that have the **User Interface** role enabled to store session data in the database. As a result, the user does not need to re-login if the load balancer redirects them to an alternative server in the case the original user interface worker is unresponsive.

On each appliance, configure the session data storage location using the **`session_store`** parameter within the advanced settings page in the Infrastructure Automation user interface:

1. Click (**Configuration**).
2. Click the **Advanced** tab.
3. Change the **`session_store`** parameter to **`sql`** in the following line (the default parameter is **`cache`**):

```
:server:
...
 :session_store: sql
```

4. Click **Save**.

Configure the `session_store` parameter to point to `sql` on each user interface appliance behind the load balancer.

Database Configuration

This section describes the Infrastructure Automation PostgreSQL database configuration. The below table provides information on each file: its location, primary function, and notes regarding behavior or recommendations.

File	Location	Description	Note

File	Location	Description	Note
`postgresql.conf`	Data Directory	Default server configuration	Adds directive to include `/etc/manageiq/postgresql.conf.d` directory
`postgresql.auto.conf`	Data Directory	Contains configuration set by the `ALTER SYSTEM` command	**Do not edit manually**
`01_miq_overrides.conf`	`/etc/manageiq/postgresql.conf.d`	Contains Infrastructure Automation default overrides	Overwritten on upgrades
<other files>	`/etc/manageiq/postgresql.conf.d`	Contains user overrides	Takes precedence if alphabetically after `01_miq_overrides.conf`

Database files

User Overrides

Store custom configurations, or user overrides, in `/etc/manageiq/postgresql.conf.d`. Name the user override file so that it follows `01_miq_overrides.conf` alphabetically. This ensures custom configurations are not overwritten on Infrastructure Automation upgrades.

The following file name example follows `01_miq_overrides.conf` alphabetically in the `/etc/manageiq/postgresql.conf.d` directory:

Example:

```
test_miq_overrides.conf
```

Reading Configuration Settings

Query the Infrastructure Automation PostgreSQL database directly to read configuration settings. See the PostgreSQL Documentation for more information.

The following example queries the Infrastructure Automation for current value set for `max_wal_senders`:

Example:

```
`psql -d vmdb_production -c 'show max_wal_senders'
```

Capacity Planning

Capacity and Utilization Collection

Infrastructure Automation server can collect and analyze capacity and utilization data from your virtual infrastructure. Use this data to understand the limitations of your current environment and plan for growth.

For some capacity and utilization data, Infrastructure Automation calculates and shows trend lines in the charts. Trend lines are created by using linear regression, which is calculated by using the capacity and utilization data that is collected by Infrastructure Automation during the interval you specify for the chart. The more data that you have the better the predictive value of the trend line.

Three server roles are associated with the collection and metric creation of capacity and utilization.

- The Capacity & Utilization Coordinator role checks to see whether it is time to collect data, somewhat like a scheduler. If it is time, a job is queued for the Capacity & Utilization Data Collector. The Coordinator role is required to complete capacity and utilization data collection. If more than one server in a specific zone has this role, only one is active at a time.
- The Capacity & Utilization Data Collector performs the actual collection of capacity and utilization data. This role has a dedicated worker, and there can be more than one server with this role in a zone.
- The Capacity & Utilization Data Processor processes all of the data that is collected, allowing Infrastructure Automation to create charts. This role has a dedicated worker, and there can be more than one server with this role in a zone.

Assigning the Capacity and Utilization Server Roles

1. Click **Configuration**, then select the server to configure from menu: **Settings > Zone** in the accordion menu on the left.
2. Browse to the **Server Roles** list in the menu: **Server > Server Control** section. From there, set the required capacity and utilization roles to **ON**, namely:
    1. **Capacity & Utilization Coordinator**
    2. **Capacity & Utilization Data Collector**
    3. **Capacity & Utilization Data Processor**
3. Click **Save**.

Data collection is enabled immediately. However, the first collection begins 5 minutes after the server is started, and every 10 minutes after that. Therefore, the longest the collection takes after enabling the Capacity & Utilization Collector role is 10 minutes. The first collection from a particular provider may take a few minutes since Infrastructure Automation is gathering data points going one month back in time.

Note:

In addition to setting the server role, you must also select which clusters and data stores to collect data for. For more information, see the **General Configuration** guide. You must have super administrator privileges to edit these settings.

Adding Database Credentials for Data Collection

After you create the new user, add the user's credentials to the settings for the provider.

1. From menu: **Compute > Infrastructure > Providers**, select an infrastructure provider to update its settings.
2. Click **Configuration**, and then **Edit Selected Infrastructure Provider**.
3. In the **Credentials** area, click **C & U Database**.
4. Type in the credentials for the new database user you created.
5. Click **Save**.
6. Restart the Capacity and Utilization Data Collector.

Data Collection for Red Hat Virtualization

To collect capacity and utilization data for Red Hat Virtualization (RHV), you must add a user to the RHV-M history database for Infrastructure Automation to use.

Perform this procedure on the PostgreSQL server where the history database (ovirt_engine_history) is located. Usually, this is the RHV-M server.

1. Using SSH, access the RHV-M database server as the root user:

   ```
   $ ssh root@example.postgres.server
   ```

2. Switch to the postgres user:

   ```
   # su - postgres
   ```

   For RHV 4.2 and 4.3, the PostgreSQL database is delivered as a software collection in version 9.5 and must be enabled first. Therefore, to run the following psql commands, you need to enable the *rh-postgresql95* collection and load into the current shell prompt by using the source command: $ source /opt/rh/rh-postgresql95/enable
3. Create the user for Infrastructure Automation to be granted read-only access to the history database:

   ```
   $ psql -U postgres -c "CREATE ROLE example_role_name WITH LOGIN ENCRYPTED PASSWORD '[password]';" -d ovirt_engine_history
   ```

4. Grant the newly created user permission to connect to the history database:

   ```
   $ psql -U postgres -c "GRANT CONNECT ON DATABASE ovirt_engine_history TO example_role_name;"
   ```

5. Grant the newly created user usage of the public schema:

   ```
   $ psql -U postgres -c "GRANT USAGE ON SCHEMA public TO example_role_name;" ovirt_engine_history
   ```

6. Generate the rest of the permissions that will be granted to the newly created user and save them to a file:

   ```
   $ psql -U postgres -c "SELECT 'GRANT SELECT ON ' || relname || ' TO example_role_name;' FROM pg_class JOIN pg_namespace ON pg_namespace.oid = pg_class.relnamespace WHERE nspname = 'public' AND relkind IN ('r', 'v', 'S');" --pset=tuples_only=on  ovirt_engine_history > grant.sql
   ```

7. Use the file that you created in the previous step to grant permissions to the newly created user:

   ```
   $ psql -U postgres -f grant.sql ovirt_engine_history
   ```

8. Remove the file that you used to grant permissions to the newly created user:

   ```
   $ rm grant.sql
   ```

9. Exit to the RHV-M database server prompt:

   ```
   $ exit
   ```

10. Update the server's firewall to accept TCP communication on port 5432:

    ```
    # firewall-cmd --add-port=5432/tcp --permanent
    ```

11. Enable external md5 authentication.

    1. For RHV 4.0, 4.1, and 4.4 update the following line in **/var/lib/pgsql/data/pg_hba.conf**:

       ```
       host    all      all    0.0.0.0/0     md5
       ```

    2. For RHV 4.2 and 4.3, update the following line in **/var/opt/rh/rh-postgresql95/lib/pgsql/data/pg_hba.conf**:

       ```
       host    all      all    0.0.0.0/0     md5
       ```

12. Enable PostgreSQL to listen for remote connections.

    1. For RHV 4.0, 4.1, and 4.4 ensure the **listen_addresses** line in **/var/lib/pgsql/data/postgresql.conf** matches the following example:

       ```
       listen_addresses  =  '*'
       ```

    2. For RHV 4.2 and 4.3, ensure the **listen_addresses** line in **/var/opt/rh/rh-postgresql95/lib/pgsql/data/postgresql.conf** matches the following example:

       ```
       listen_addresses  =  '*'
       ```

13. Reload the PostgreSQL configuration.

    1. For RHV 4.0, 4.1, and 4.4:

       ```
       # systemctl reload postgresql
       ```

    2. For RHV 4.2 and 4.3:

       ```
       # systemctl reload rh-postgresql95-postgresql
       ```

Data Collection for Red Hat Enterprise Linux OpenStack Platform

Before you can collect data from a Red Hat Enterprise Linux OpenStack Platform (RHEL-OSP) provider, you must install Ceilometer and configure it to accept queries from external systems.

These instructions require a Red Hat Enterprise Linux 6.4 @base installation of RHEL-OSP and registration to a satellite that has access to both the **RHEL-OSP** and **RHEL Server Optional** channels. Perform all steps on your RHEL-OSP system.

1. Add the required channels and update your system:

   ```
   # rhn-channel --add -c rhel-x86_64-server-6-ost-3 -c rhel-x86_64-server-optional-6
   # yum update -y
   # reboot
   ```

2. Install **Ceilometer**:

   ```
   # yum install *ceilometer*
   ```

3. Install and start the MongoDB store:

   ```
   # yum install mongodb-server
   # sed -i '/--smallfiles/!s/OPTIONS=\"/OPTIONS=\"--smallfiles /' /etc/sysconfig/mongod
   # service mongod start
   ```

4. Create the following users and roles:

   ```
   # SERVICE_TENANT=$(keystone tenant-list | grep services | awk '{print $2}')
   # ADMIN_ROLE=$(keystone role-list | grep ' admin ' | awk '{print $2}')
   # SERVICE_PASSWORD=servicepass
   # CEILOMETER_USER=$(keystone user-create --name=ceilometer \
   --pass="$SERVICE_PASSWORD" \
   --tenant_id $SERVICE_TENANT \
   --email=ceilometer@example.com | awk '/ id / {print $4}')
   # RESELLER_ROLE=$(keystone role-create --name=ResellerAdmin | awk '/ id / {print $4}')
   # ADMIN_ROLE=$(keystone role-list | awk '/ admin / {print $2}')
   # for role in $RESELLER_ROLE $ADMIN_ROLE ; do
   keystone user-role-add --tenant_id $SERVICE_TENANT \
   --user_id $CEILOMETER_USER --role_id $role
   done
   ```

5. Configure the authtoken in **ceilometer.conf**:

   ```
   # openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_host 127.0.0.1
   # openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_port 35357
   # openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken auth_protocol http
   # openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken admin_tenant_name services
   # openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken admin_user ceilometer
   ```

```
 # openstack-config --set /etc/ceilometer/ceilometer.conf keystone_authtoken admin_password
$SERVICE_PASSWORD
```

6. Configure the user credentials in **ceilometer.conf**:

```
 # openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT os_auth_url
http://127.0.0.1:35357/v2.0
 # openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT os_tenant_name services
 # openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT os_password $SERVICE_PASSWORD
 # openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT os_username ceilometer
```

7. Start the Ceilometer services:

```
 # for svc in compute central collector api ; do
   service openstack-ceilometer-$svc start
   done
```

8. Register an endpoint with the service catalog. Replace **$EXTERNALIFACE** with the IP address of your external interface:

```
 # keystone service-create --name=ceilometer \
 --type=metering --description="Ceilometer Service"
 # CEILOMETER_SERVICE=$(keystone service-list | awk '/ceilometer/ {print $2}')
 # keystone endpoint-create \
 --region RegionOne \
 --service_id $CEILOMETER_SERVICE \
 --publicurl "http://$EXTERNALIFACE:8777/" \
 --adminurl "http://$EXTERNALIFACE:8777/" \
 --internalurl "http://localhost:8777/"
```

9. Enable access to Ceilometer from external systems:

```
 # iptables -I INPUT -p tcp -m multiport --dports 8777 -m comment --comment "001 ceilometer incoming" -j
ACCEPT
 # iptables save
```

10. Confirm the status of OpenStack and the Ceilometer services:

```
# openstack-status
# for svc in compute central collector api ; do
  service openstack-ceilometer-$svc status
  done
```

11. Verify that Ceilometer is working correctly by authenticating as a user with running instances, for example **admin**. Pipe the sample for the CPU meter to count lines, and confirm that the value changes according to the interval specified in **/etc/ceilometer/pipeline.yaml**. The default interval is 600 seconds.

```
# . ~/keystonerc_admin
# ceilometer sample-list -m cpu |wc -l
```

12. Add the configured OpenStack provider to Infrastructure Automation. See After adding the provider, capacity, and utilization data for your instances populate in a few minutes.

Capacity and Utilization Data Collected

Infrastructure Automation generates charts from the collected data that can be used to plan your hardware and virtual machine needs. Depending on the type of data, these charts can include lines for averages, maximums, minimums, and trends.

Note:

For reporting of daily capacity and utilization data, incomplete days (days with less than 24 hourly data points from midnight to midnight) that are at the beginning or end of the requested interval are excluded. Days with less than 24 hourly data points would be inaccurate and including them would skew trend lines. Therefore, at least one full day of hourly data from midnight to midnight is necessary for displaying the capacity and utilization charts under the menu: **Compute > Infrastructure** tab.

Capacity and Utilization Charts for Hosts, Clusters, and Virtual Machines

Resource Type	CPU Usage	CPU States	Disk I/O	Memory Usage	Network I/O	Running VMS	Running Hosts
Host	Y	Y	Y	Y	Y	Y	NA
Cluster	Y	Y	Y	Y	Y	Y	Y
Virtual Machine	Y	Y	Y	Y	Y	NA	NA

Capacity and Utilization Charts for Hosts, Clusters, and Virtual Machines

Capacity and Utilization Charts for Datastores

Charts that are created include:

Space by VM Type	Virtual Machines and Hosts
Used Space	Number of VMs by Type
Disk files Space	Hosts
Snapshot Files Space	Virtual Machines
Memory Files Space	
Non-VM Files	
Used Disk Space	

Capacity and Utilization Charts for Datastores

Capacity and Utilization Chart Features

Capacity and utilization charts for host, clusters, virtual machines, and datastore provides its own set of special features that include zooming in on a chart and shortcut menus.

Zooming into a Chart

1. Browse to the chart you want to zoom. If you hover anywhere on the chart, two dashed lines appear to target a coordinate of the chart.
2. Click (**Click to zoom in**) in the lower left corner of the chart to zoom into it.
3. To go back to the regular view click (**Click to zoom out**) on the enlarged chart.

Drilling into Chart Data

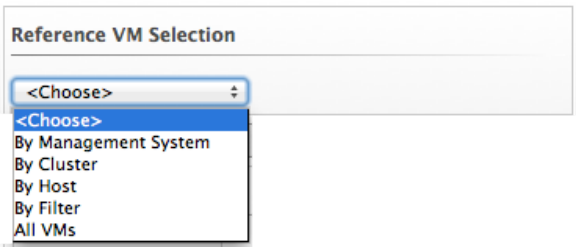
1. Browse to the chart you want to get more detail from.
2. Hover over a data point to see the coordinates.
3. Click a data point to open a shortcut menu for the chart. In this example, you can use the shortcut menu to go to the hourly chart or display the virtual machines that were running at the time the data was captured.
    - If you are viewing the **CPU**, **Disk**, **Memory**, or **Network** charts, selecting from the **Chart** option will change all of the charts on the page to the new interval selected.
    - If you are viewing the **CPU**, **Disk**, **Memory**, or **Network** charts, selecting from the **Display** option will allow you to drill into the virtual machines or **Hosts** that were running at the time.
    - If you are viewing the **VM** or **Hosts** chart, the **Display** menu allows you to view running or stopped virtual machines. The time of the data point will be displayed in addition to the virtual machines that apply. From here, click a virtual machine to view its details.

Planning Where to Put a New Virtual Machine

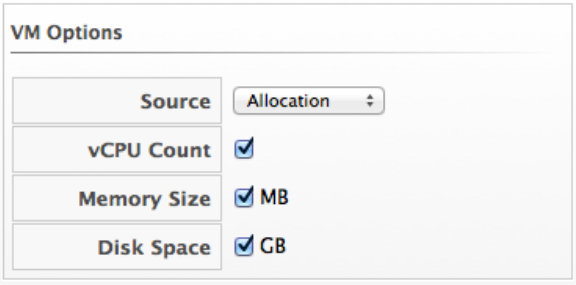
You can use the data that is collected in the VMDB to plan where you can put more virtual machines. With Infrastructure Automation, you can use a reference virtual machine as an example to plan on which hosts and clusters you can place a new virtual machine.

** Note:** For more information about utilization trends and identifying bottlenecks in your environment, see Data Optimization in Managing Infrastructure and Inventory.

1. Browse to **Optimize > Planning**.
2. From **Reference VM Selection**, use the drop-down menu to select the virtual machine that is most like the one that you want to add.

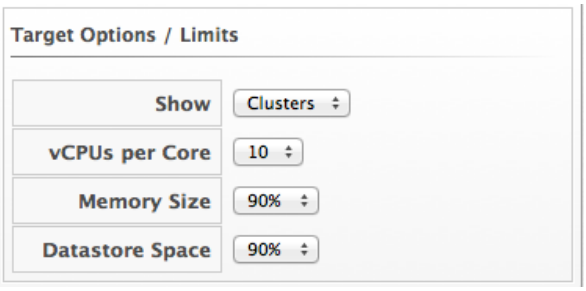


3. Select the required VM Options for what you want to base the calculations on.



From the **Source** list, select the type of data to use as the source for your projections. For example, select **Allocation** to calculate based on the current allocation values of each resource (CPU, memory, or disk space) for the reference virtual machine. Use **Reservation** to project based on the current guaranteed value of the specific resource (CPU Speed, CPU count, memory, or disk space) although that amount might not be allocated to the virtual machine at a specific moment in time. Select **Usage** if you want to calculate based on usage history of the reference virtual machine. Use **Manual Input** to enter your own set of parameters for each resource.

4. From **Target Options / Limits**, select if you want to use clusters or hosts as your targets.



Also, select the limit of how high the projection can go for CPU, memory, and datastore space. If you are targeting hosts, you are able to select a filter for which hosts can be targets.

5. From Trend Options, select how far back you want to use the trend data for, a Time Profile and Time Zone if applicable. Note that Time Profile will only show if the logged on user has a Time Profile available.
6. Click Submit.

The **Summary** tab shows the best clusters or hosts on which to place the virtual machines. The **Report** tab shows the best fit and statistics on the reference virtual machine in a tabular format. From the **Report** tab, you can also create a PDF of the report or download the data in **`txt`** or **`csv`** format.

Appendix

Example PostgreSQL Configuration File

```
# -----------------------------
# PostgreSQL configuration file - MIQ Dedicated Appliance Configuration
# -----------------------------
#
# This file consists of lines of the form:
#
#   name = value
#
# (The "=" is optional.)  Whitespace may be used.  Comments are introduced with
# "#" anywhere on a line.  The complete list of parameter names and allowed
# values can be found in the PostgreSQL documentation.
#
# The commented-out settings shown in this file represent the default values.
# Re-commenting a setting is NOT sufficient to revert it to the default value;
# you need to reload the server.
#
# This file is read on server startup and when the server receives a SIGHUP
# signal.  If you edit the file on a running system, you have to SIGHUP the
# server for the changes to take effect, or use "pg_ctl reload".  Some
# parameters, which are marked below, require a server shutdown and restart to
# take effect.
#
# Any parameter can also be given as a command-line option to the server, e.g.,
# "postgres -c log_connections=on".  Some parameters can be changed at run time
# with the "SET" SQL command.
```

```
#
# Memory units:  kB = kilobytes        Time units:  ms  = milliseconds
#                MB = megabytes                     s   = seconds
#                GB = gigabytes                     min = minutes
#                                                   h   = hours
#                                                   d   = days


#------------------------------------------------------------------------------
# FILE LOCATIONS
#------------------------------------------------------------------------------

# The default values of these variables are driven from the -D command-line
# option or PGDATA environment variable, represented here as ConfigDir.

#data_directory = 'ConfigDir'       # use data in another directory
                    # (change requires restart)
#hba_file = 'ConfigDir/pg_hba.conf' # host-based authentication file
                    # (change requires restart)
#ident_file = 'ConfigDir/pg_ident.conf' # ident configuration file
                    # (change requires restart)

# If external_pid_file is not explicitly set, no extra PID file is written.
#external_pid_file = '(none)'       # write an extra PID file
                    # (change requires restart)


#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------

# - Connection Settings -

listen_addresses = '10.132.50.128'  # MIQ Value;
#listen_addresses = 'localhost'     # what IP address(es) to listen on;
                    # comma-separated list of addresses;
                    # defaults to 'localhost', '*' = all
                    # (change requires restart)
#port = 5432                # (change requires restart)
max_connections = 1600          # MIQ Value increased
#max_connections = 100          # (change requires restart) Note:  Increasing max_connections costs ~400
bytes of shared memory per connection slot, plus lock space (see max_locks_per_transaction).
#superuser_reserved_connections = 3 # (change requires restart)
#unix_socket_directory = ''     # (change requires restart)
#unix_socket_group = ''         # (change requires restart)
#unix_socket_permissions = 0777     # begin with 0 to use octal notation
                    # (change requires restart)
#bonjour = off              # advertise server via Bonjour
                    # (change requires restart)
#bonjour_name = ''          # defaults to the computer name
                    # (change requires restart)

# - Security and Authentication -

#authentication_timeout = 1min      # 1s-600s
#ssl = off              # (change requires restart)
#ssl_ciphers = 'ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH'  # allowed SSL ciphers
                    # (change requires restart)
#ssl_renegotiation_limit = 512MB    # amount of data between renegotiations
#password_encryption = on
#db_user_namespace = off

# Kerberos and GSSAPI
#krb_server_keyfile = ''
#krb_srvname = 'postgres'       # (Kerberos only)
#krb_caseins_users = off

# - TCP Keepalives -
# see "man 7 tcp" for details

tcp_keepalives_idle = 3         # MIQ Value;
#tcp_keepalives_idle = 0        # TCP_KEEPIDLE, in seconds;
                    # 0 selects the system default
tcp_keepalives_interval = 75        # MIQ Value;
#tcp_keepalives_interval = 0        # TCP_KEEPINTVL, in seconds;
                    # 0 selects the system default
tcp_keepalives_count = 9        # MIQ Value;
#tcp_keepalives_count = 0       # TCP_KEEPCNT;
                    # 0 selects the system default
```

```
#------------------------------------------------------------------------------
# RESOURCE USAGE (except WAL)
#------------------------------------------------------------------------------

# - Memory -

#shared_buffers = 128MB         # MIQ Value SHARED CONFIGURATION
shared_buffers = 4GB            # MIQ Value DEDICATED CONFIGURATION increased
#shared_buffers = 32MB          # min 128kB
                    # (change requires restart)
#temp_buffers = 8MB         # min 800kB
#max_prepared_transactions = 0      # zero disables the feature
                    # (change requires restart)
# Note:  Increasing max_prepared_transactions costs ~600 bytes of shared memory
# per transaction slot, plus lock space (see max_locks_per_transaction).
# It is not advisable to set max_prepared_transactions nonzero unless you
# actively intend to use prepared transactions.
#work_mem = 1MB             # min 64kB
#maintenance_work_mem = 16MB        # min 1MB
#max_stack_depth = 2MB          # min 100kB

# - Kernel Resource Usage -

#max_files_per_process = 1000       # min 25
                    # (change requires restart)
#shared_preload_libraries = ''      # (change requires restart)

# - Cost-Based Vacuum Delay -

#vacuum_cost_delay = 0ms        # 0-100 milliseconds
#vacuum_cost_page_hit = 1       # 0-10000 credits
#vacuum_cost_page_miss = 10     # 0-10000 credits
#vacuum_cost_page_dirty = 20        # 0-10000 credits
#vacuum_cost_limit = 200        # 1-10000 credits

# - Background Writer -

#bgwriter_delay = 200ms         # 10-10000ms between rounds
#bgwriter_lru_maxpages = 100        # 0-1000 max buffers written/round
#bgwriter_lru_multiplier = 2.0      # 0-10.0 multipler on buffers scanned/round

# - Asynchronous Behavior -

#effective_io_concurrency = 1       # 1-1000. 0 disables prefetching


#------------------------------------------------------------------------------
# WRITE AHEAD LOG
#------------------------------------------------------------------------------

# - Settings -

#wal_level = minimal            # minimal, archive, or hot_standby
                    # (change requires restart)
#fsync = on             # turns forced synchronization on or off
#synchronous_commit = on        # synchronization level; on, off, or local
#wal_sync_method = fsync        # the default is the first option
                    # supported by the operating system:
                    #   open_datasync
                    #   fdatasync (default on Linux)
                    #   fsync
                    #   fsync_writethrough
                    #   open_sync
#full_page_writes = on          # recover from partial page writes
wal_buffers = 16MB          # MIQ Value;
#wal_buffers = -1           # min 32kB, -1 sets based on shared_buffers
                    # (change requires restart)
#wal_writer_delay = 200ms       # 1-10000 milliseconds

#commit_delay = 0           # range 0-100000, in microseconds
#commit_siblings = 5            # range 1-1000

# - Checkpoints -

checkpoint_segments = 15        # MIQ Value;
#checkpoint_segments = 3        # in logfile segments, min 1, 16MB each
#checkpoint_timeout = 5min      # range 30s-1h
checkpoint_completion_target = 0.9  # MIQ Value;
#checkpoint_completion_target = 0.5 # checkpoint target duration, 0.0 - 1.0
#checkpoint_warning = 30s       # 0 disables
```

```
# - Archiving -

#archive_mode = off     # allows archiving to be done
                # (change requires restart)
#archive_command = ''       # command to use to archive a logfile segment
#archive_timeout = 0        # force a logfile segment switch after this
                # number of seconds; 0 disables


#------------------------------------------------------------------------------
# REPLICATION
#------------------------------------------------------------------------------

# - Master Server -

# These settings are ignored on a standby server

#max_wal_senders = 0        # max number of walsender processes
                # (change requires restart)
#wal_sender_delay = 1s      # walsender cycle time, 1-10000 milliseconds
#wal_keep_segments = 0      # in logfile segments, 16MB each; 0 disables
#vacuum_defer_cleanup_age = 0   # number of xacts by which cleanup is delayed
#replication_timeout = 60s  # in milliseconds; 0 disables
#synchronous_standby_names = '' # standby servers that provide sync rep
                # comma-separated list of application_name
                # from standby(s); '*' = all

# - Standby Servers -

# These settings are ignored on a master server

#hot_standby = off          # "on" allows queries during recovery
                    # (change requires restart)
#max_standby_archive_delay = 30s    # max delay before canceling queries
                    # when reading WAL from archive;
                    # -1 allows indefinite delay
#max_standby_streaming_delay = 30s  # max delay before canceling queries
                    # when reading streaming WAL;
                    # -1 allows indefinite delay
#wal_receiver_status_interval = 10s # send replies at least this often
                    # 0 disables
#hot_standby_feedback = off     # send info from standby to prevent
                    # query conflicts


#------------------------------------------------------------------------------
# QUERY TUNING
#------------------------------------------------------------------------------

# - Planner Method Configuration -

#enable_bitmapscan = on
#enable_hashagg = on
#enable_hashjoin = on
#enable_indexscan = on
#enable_material = on
#enable_mergejoin = on
#enable_nestloop = on
#enable_seqscan = on
#enable_sort = on
#enable_tidscan = on

# - Planner Cost Constants -

#seq_page_cost = 1.0            # measured on an arbitrary scale
#random_page_cost = 4.0         # same scale as above
#cpu_tuple_cost = 0.01          # same scale as above
#cpu_index_tuple_cost = 0.005       # same scale as above
#cpu_operator_cost = 0.0025     # same scale as above
#effective_cache_size = 128MB

# - Genetic Query Optimizer -

#geqo = on
#geqo_threshold = 12
#geqo_effort = 5            # range 1-10
#geqo_pool_size = 0         # selects default based on effort
#geqo_generations = 0           # selects default based on effort
#geqo_selection_bias = 2.0      # range 1.5-2.0
#geqo_seed = 0.0            # range 0.0-1.0
```

```
# - Other Planner Options -

#default_statistics_target = 100    # range 1-10000
#constraint_exclusion = partition   # on, off, or partition
#cursor_tuple_fraction = 0.1        # range 0.0-1.0
#from_collapse_limit = 8
#join_collapse_limit = 8        # 1 disables collapsing of explicit
                    # JOIN clauses


#------------------------------------------------------------------------------
# ERROR REPORTING AND LOGGING
#------------------------------------------------------------------------------

# - Where to Log -


log_destination = 'stderr'      # Valid values are combinations of
                    # stderr, csvlog, syslog, and eventlog,
                    # depending on platform.  csvlog
                    # requires logging_collector to be on.

# This is used when logging to stderr:
logging_collector = on      # Enable capturing of stderr and csvlog
                    # into log files. Required to be on for
                    # csvlogs.
                    # (change requires restart)

# These are only used if logging_collector is on:
log_directory = '/www/postgres/log' # Customer specific setting
#log_directory = 'log'       # directory where log files are written,
                    # can be absolute or relative to PGDATA
log_filename = 'postgresql-%Y-%m-%d.log'        # log file name pattern,
                    # can include strftime() escapes
log_file_mode = 0644                    # creation mode for log files,
                    # begin with 0 to use octal notation
log_truncate_on_rotation = on           # If on, an existing log file with the
                    # same name as the new log file will be
                    # truncated rather than appended to.
                    # But such truncation only occurs on
                    # time-driven rotation, not on restarts
                    # or size-driven rotation.  Default is
                    # off, meaning append to existing files
                    # in all cases.
log_rotation_age = 1d                   # Automatic rotation of logfiles will
                    # happen after that time.  0 disables.
log_rotation_size = 0                   # Automatic rotation of logfiles will
                    # happen after that much log output.
                    # 0 disables.

# These are relevant when logging to syslog:
#syslog_facility = 'LOCAL0'
#syslog_ident = 'postgres'

#silent_mode = off          # Run server silently.
                    # DO NOT USE without syslog or
                    # logging_collector
                    # (change requires restart)


# - When to Log -

#client_min_messages = notice       # values in order of decreasing detail:
                    #   debug5
                    #   debug4
                    #   debug3
                    #   debug2
                    #   debug1
                    #   log
                    #   notice
                    #   warning
                    #   error

#log_min_messages = warning     # values in order of decreasing detail:
                    #   debug5
                    #   debug4
                    #   debug3
                    #   debug2
                    #   debug1
                    #   info
                    #   notice
```

```
                    #   warning
                    #   error
                    #   log
                    #   fatal
                    #   panic

#log_min_error_statement = error    # values in order of decreasing detail:
                    #   debug5
                    #   debug4
                    #   debug3
                    #   debug2
                    #   debug1
                    #   info
                    #   notice
                    #   warning
                    #   error
                    #   log
                    #   fatal
                    #   panic (effectively off)

log_min_duration_statement = 5000   # MIQ Value- ANY statement > 5 seconds
#log_min_duration_statement = -1    # -1 is disabled, 0 logs all statements
                    # and their durations, > 0 logs only
                    # statements running at least this number
                    # of milliseconds


# - What to Log -

#debug_print_parse = off
#debug_print_rewritten = off
#debug_print_plan = off
#debug_pretty_print = on
#log_checkpoints = off
#log_connections = off
#log_disconnections = off
#log_duration = off
#log_error_verbosity = default      # terse, default, or verbose messages
#log_hostname = off
log_line_prefix = '%t:%r:%c:%u@%d:[%p]:'    # MIQ Value;
#log_line_prefix = ''           # special values:
                    #   %a = application name
                    #   %u = user name
                    #   %d = database name
                    #   %r = remote host and port
                    #   %h = remote host
                    #   %p = process ID
                    #   %t = timestamp without milliseconds
                    #   %m = timestamp with milliseconds
                    #   %i = command tag
                    #   %e = SQL state
                    #   %c = session ID
                    #   %l = session line number
                    #   %s = session start timestamp
                    #   %v = virtual transaction ID
                    #   %x = transaction ID (0 if none)
                    #   %q = stop here in non-session
                    #        processes
                    #   %% = '%'
                    # e.g. '<%u%%%d> '
log_lock_waits = on          # MIQ Value - used to track possible deadlock issues
#log_lock_waits = off           # log lock waits >= deadlock_timeout
#log_statement = 'none'         # none, ddl, mod, all
#log_temp_files = -1            # log temporary files equal or larger
                    # than the specified size in kilobytes;
                    # -1 disables, 0 logs all temp files
#log_timezone = '(defaults to server environment setting)'


#------------------------------------------------------------------------------
# RUNTIME STATISTICS
#------------------------------------------------------------------------------

# - Query/Index Statistics Collector -

#track_activities = on
track_counts = on           # MIQ Value;
#track_counts = on
#track_functions = none         # none, pl, all
#track_activity_query_size = 1024   # (change requires restart)
#update_process_title = on
```

```
#stats_temp_directory = 'pg_stat_tmp'


# - Statistics Monitoring -

#log_parser_stats = off
#log_planner_stats = off
#log_executor_stats = off
#log_statement_stats = off


#------------------------------------------------------------------------------
# AUTOVACUUM PARAMETERS
#------------------------------------------------------------------------------

autovacuum = on             # MIQ Value;
#autovacuum = on            # Enable autovacuum subprocess?  'on'
                    # requires track_counts to also be on.
log_autovacuum_min_duration = 0     # MIQ Value;
#log_autovacuum_min_duration = -1   # -1 disables, 0 logs all actions and
                    # their durations, > 0 logs only
                    # actions running at least this number
                    # of milliseconds.
autovacuum_max_workers = 1      # max number of autovacuum subprocesses
                    # (change requires restart)
autovacuum_naptime = 30min      # MIQ Value;
#autovacuum_naptime = 1min      # time between autovacuum runs
autovacuum_vacuum_threshold = 500   # MIQ Value;
#autovacuum_vacuum_threshold = 50   # min number of row updates before
                    # vacuum
autovacuum_analyze_threshold = 500  # MIQ Value;
#autovacuum_analyze_threshold = 50  # min number of row updates before
                    # analyze
autovacuum_vacuum_scale_factor = 0.05   # MIQ Value;
#autovacuum_vacuum_scale_factor = 0.2   # fraction of table size before vacuum
#autovacuum_analyze_scale_factor = 0.1  # fraction of table size before analyze
#autovacuum_freeze_max_age = 200000000  # maximum XID age before forced vacuum
                    # (change requires restart)
#autovacuum_vacuum_cost_delay = 20ms    # default vacuum cost delay for
                    # autovacuum, in milliseconds;
                    # -1 means use vacuum_cost_delay
#autovacuum_vacuum_cost_limit = -1  # default vacuum cost limit for
                    # autovacuum, -1 means use
                    # vacuum_cost_limit


#------------------------------------------------------------------------------
# CLIENT CONNECTION DEFAULTS
#------------------------------------------------------------------------------

# - Statement Behavior -

#search_path = '"$user",public'     # schema names
#default_tablespace = ''        # a tablespace name, '' uses the default
#temp_tablespaces = ''          # a list of tablespace names, '' uses
                    # only default tablespace
#check_function_bodies = on
#default_transaction_isolation = 'read committed'
#default_transaction_read_only = off
#default_transaction_deferrable = off
#session_replication_role = 'origin'
#statement_timeout = 0          # in milliseconds, 0 is disabled
#statement_timeout = 43200000           # MIQ statment timeout of 12 hours as a default
#vacuum_freeze_min_age = 50000000
#vacuum_freeze_table_age = 150000000
#bytea_output = 'hex'           # hex, escape
#xmlbinary = 'base64'
#xmloption = 'content'

# - Locale and Formatting -

datestyle = 'iso, mdy'
#intervalstyle = 'postgres'
#timezone = '(defaults to server environment setting)'
#timezone_abbreviations = 'Default'     # Select the set of available time zone
                    # abbreviations.  Currently, there are
                    #   Default
                    #   Australia
                    #   India
                    # You can create your own file in
                    # share/timezonesets/.
```

```
#extra_float_digits = 0         # min -15, max 3
#client_encoding = sql_ascii        # actually, defaults to database
                    # encoding

# These settings are initialized by initdb, but they can be changed.
lc_messages = 'en_US.UTF-8'         # locale for system error message
                    # strings
lc_monetary = 'en_US.UTF-8'         # locale for monetary formatting
lc_numeric = 'en_US.UTF-8'          # locale for number formatting
lc_time = 'en_US.UTF-8'             # locale for time formatting

# default configuration for text search
default_text_search_config = 'pg_catalog.english'

# - Other Defaults -

#dynamic_library_path = '$libdir'
#local_preload_libraries = ''


#------------------------------------------------------------------------------
# LOCK MANAGEMENT
#------------------------------------------------------------------------------

deadlock_timeout = 5s           # MIQ Value - one second is too low, 5 seconds is more "interesting"
#deadlock_timeout = 1s
#max_locks_per_transaction = 64     # min 10
                    # (change requires restart)
# Note:  Each lock table slot uses ~270 bytes of shared memory, and there are
# max_locks_per_transaction * (max_connections + max_prepared_transactions)
# lock table slots.
#max_pred_locks_per_transaction = 64    # min 10
                    # (change requires restart)

#------------------------------------------------------------------------------
# VERSION/PLATFORM COMPATIBILITY
#------------------------------------------------------------------------------

# - Previous PostgreSQL Versions -

#array_nulls = on
#backslash_quote = safe_encoding    # on, off, or safe_encoding
#default_with_oids = off
escape_string_warning = off     # MIQ Value no sure why this is enabled
#escape_string_warning = on
#lo_compat_privileges = off
#quote_all_identifiers = off
#sql_inheritance = on
standard_conforming_strings = off   # MIQ Value not sure why this is enabled
#standard_conforming_strings = on
#synchronize_seqscans = on

# - Other Platforms and Clients -

#transform_null_equals = off


#------------------------------------------------------------------------------
# ERROR HANDLING
#------------------------------------------------------------------------------

#exit_on_error = off                # terminate session on any error?
#restart_after_crash = on           # reinitialize after backend crash?


#------------------------------------------------------------------------------
# CUSTOMIZED OPTIONS
#------------------------------------------------------------------------------

#custom_variable_classes = ''       # list of custom variable class names
```


- Integration with OpenShift Container Platform
  - Overview
  - Prerequisites
  - Retrieving the OpenShift Container Platform Management Token
  - Configuring Infrastructure Automation
  - Configuring Infrastructure Automation Capacity and Utilization
    - Enabling SmartState Analysis
  - Adding OpenShift Container Platform as a Container Provider

- Container Image Scanning
- Configuring Image Scanning
- Scheduling A Recurring Scan
- Working with Images
  - Viewing Results
  - Manual Scanning
  - Evaluating Compliance
  - Generating a Report on Images
- OpenSCAP Policy Profile
  - Customizing the Scanning Policy Profile
- Controlling OpenShift Pod Execution
- Reference

Integration with OpenShift Container Platform

Overview

This guide walks you through adding a Red Hat OpenShift Container Platform cluster to a Infrastructure Automation container provider catalog. This deployment focuses on enabling the OpenShift Container Platform cluster metrics plug-in, so that Infrastructure Automation can collect information from an OpenShift Container Platform cluster upon integration.

Each procedure in this guide is covered in greater detail in the Infrastructure Automation and OpenShift Container Platform product documentation. However, links to the corresponding sections are provided for more detail.

The following sections will describe the required configuration for both products prior to integration.

Prerequisites

This guide assumes that you have:

- Already deployed Infrastructure Automation on your chosen platform
- Already deployed OpenShift Container Platform

When enabling metrics on OpenShift Container Platform, you can store your metrics data on *persistent* or *non-persistent* storage. To use persistent storage, you need to provision a persistent volume specifically for this purpose before configuring the metrics components. See Persistent Volumes in the OpenShift Container Platform *Architecture* documentation for more information.

Retrieving the OpenShift Container Platform Management Token

After enabling cluster metrics on your OpenShift Container Platform deployment, retrieve the *management token* while you are still logged in to the OpenShift Container Platform host. This will be required later in Adding OpenShift Container Platform as a Container Provider.

Run the following to obtain the token needed to add a Red Hat OpenShift provider:

1. Obtain the **`management`** service account token name:

```
# oc describe sa -n management-infra management-admin
...
Tokens:  management-admin-token-0f3fh
         management-admin-token-q7a87
```

2. Select and describe one of the tokens to retrieve the full token output, replacing **`management-admin-token-0f3fh`** with the name of your token:

```
# oc describe secret -n management-infra management-admin-token-0f3fh
...
Data
====
token:  eyJhbGciOiJSUzI1NiI...
```

Configuring Infrastructure Automation

Configuring Infrastructure Automation involves two steps:

1. Configuring Infrastructure Automation Capacity and Utilization, and

2. Enabling SmartState Analysis

These steps are required to allow Infrastructure Automation to collect metrics from OpenShift Container Platform Enabling OpenShift Container Platform Metrics and use them to perform a SmartState analysis. You can choose different servers to perform either function; the following sections assume that you will.

Configuring Infrastructure Automation Capacity and Utilization

For metrics collection to work properly, you also need to configure Infrastructure Automation to allow for all three **Capacity & Utilization** server roles, which are available from the settings menu under **Configuration > Server > Server Control**.

To enable these server roles:

1. Click **Configuration**, then select the server to configure from menu: **Settings > Zone** in the accordion menu on the left.
2. Browse to the **Server Roles** list in the menu: **Server > Server Control** section. From there, set the required capacity and utilization roles to **ON**, namely:
    1. **Capacity & Utilization Coordinator**
    2. **Capacity & Utilization Data Collector**
    3. **Capacity & Utilization Data Processor**
3. Click **Save**.

Data collection is enabled immediately. However, the first collection begins 5 minutes after the server is started, and every 10 minutes after that. Therefore, the longest the collection takes after enabling the Capacity & Utilization Collector role is 10 minutes. The first collection from a particular provider may take a few minutes since Infrastructure Automation is gathering data points going one month back in time.

For more information, see Capacity and Utilization Collection in the *Deployment Planning Guide*.

Enabling SmartState Analysis

After enabling the required server roles, enable SmartState analysis. See Smart State Analysis Support from the Support Matrix.

Enabling SmartState analysis is similar to Assigning the Capacity and Utilization Server Roles, in that the procedure also involves enabling server roles on a specific server. To do so:

1. Click **Configuration**.
2. Select the server to configure from menu: **Settings**[Zone] in the left pane of the appliance.
3. Browse to the **Server Roles** list in the menu: **Server > Server Control** section. From there, set the appropriate SmartState roles to **ON**. Namely:
    1. **SmartProxy**
    2. **SmartState Analysis**
4. Click **Save**.

Adding OpenShift Container Platform as a Container Provider

At this point, you should now be ready to add your OpenShift cluster to Infrastructure Automation as a container provider. To do so, prepare the token you retrieved in Retrieving the OpenShift Container Platform Management Token and follow the procedure below:

1. Browse to menu: **Compute > Containers > Providers**.
2. Click **Configuration**, then click (**Add a New Containers Provider**).
3. Enter a **Name** for the provider.
4. From the **Type** list, select **OpenShift Container Platform**.
5. Enter the appropriate **Zone** for the provider. If you do not specify a zone, it is set to `default`.
6. In the **Default** tab, configure the following for the OpenShift provider:
    1. Select a **Security Protocol** method to specify how to authenticate the provider:
        - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.

- **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.

  Note:

  You can obtain your OpenShift Container Platform provider's CA certificate for all endpoints (default, metrics, alerts) from **`/etc/origin/master/ca.crt`**. Paste the output (a block of text starting with **`-----BEGIN CERTIFICATE-----`**) into the **Trusted CA Certificates** field.

- **SSL without validation**: Authenticate the provider insecurely (not recommended).

2. Enter the **Hostname** (or IPv4 or IPv6 address) of the provider.

   Important:

   The **Hostname** must use a unique fully qualified domain name.

3. Enter the **API Port** of the provider. The default port is **`6443`**.

4. Enter a token for your provider in the **Token** box.

   Note:

   To obtain a token for your provider, run the **`oc get secret`** command on your provider; see Obtaining an OpenShift Container Platform Management Token.

   For example:

   # oc get secret –namespace management-infra management-admin-token-8ixxs –template='' | base64 –decode

5. Click **Validate** to confirm that Infrastructure Automation can connect to the OpenShift Container Platform provider.

7. In the **Metrics** endpoint tab optionally configure the metrics service details:

   1. Select a service type (e.g.: Prometheus) or leave disabled

   2. Select a **Security Protocol** method to specify how to authenticate the service:

      - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.

      - **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.

      - **SSL without validation**: Authenticate the provider insecurely using SSL. (Not recommended)

   3. Enter the **Hostname** (or IPv4 or IPv6 address) of the metrics service.

      In order to find the hostname you can use the **`oc get route`** command.

      You can retrieve the Prometheus route by running:

      ```
      oc get route prometheus-k8s -n openshift-monitoring
      ```

      Example:

      ```
      oc get route prometheus-k8s -n openshift-monitoring
      NAME              HOST/PORT                                          PATH              SERVICES
      PORT      TERMINATION          WILDCARD
      prometheus-k8s    prometheus-k8s-openshift-monitoring.ocp.example.com  prometheus-k8s    web
      reencrypt/Redirect   None
      ```

      In this example you would use **`prometheus-k8s-openshift-monitoring.ocp.example.com`** as the metrics hostname.

   4. Enter the **API Port** of your metrics endpoint uses a non-standard port for access. The default port is **`443`**.

   5. Click **Validate** to confirm that Infrastructure Automation can connect to the metrics endpoint.

8. In the **Virtualization** endpoint tab optionally configure the virtualization service details:

   1. Select a service type (OpenShift Virtualization / KubeVirt) or leave disabled

   2. Select a **Security Protocol** method to specify how to authenticate the service:

      - **SSL**: Authenticate the provider securely using a trusted Certificate Authority. Select this option if the provider has a valid SSL certificate and it is signed by a trusted Certificate Authority. No further configuration is required for this option.

- **SSL trusting custom CA**: Authenticate the provider with a self-signed certificate. For this option, copy your provider's CA certificate to the **Trusted CA Certificates** box in PEM format.
- **SSL without validation**: Authenticate the provider insecurely using SSL. (Not recommended)

3. Enter the **Hostname** (or IPv4 or IPv6 address) of the virtualization service.
4. Enter the **API Port** of your virtualization service. The default port is **`443`**.
5. Enter a token for your provider in the **Token** box.

9. Click the **Advanced** tab to add image inspector settings for scanning container images on your provider using OpenSCAP.

   Note:

   - These settings control downloading the image inspector container image from the registry and obtaining the Common Vulnerabilities and Exposures (CVE) information (for effective scanning) via a proxy.
   - CVE URL that Infrastructure Automation requires to be open for OpenSCAP scanning: https://www.redhat.com/security/data/metrics/ds/. This information is based on the source code of OpenSCAP.

   1. Enter the proxy information for the provider in either **HTTP**, **HTTPS**, or **NO Proxy** depending on your environment.
   2. Enter the **Image-Inspector Repository** information. For example, **`openshift3/image-inspector`**.
   3. Enter the **Image-Inspector Registry** information. For example, **`registry.access.redhat.com`**.
   4. Enter the **Image-Inspector Tag** value. A tag is a mark used to differentiate images in a repository, typically by the application version stored in the image.
   5. Enter **`https://www.redhat.com/security/data/metrics/ds/`** in **CVE location**.

10. Click **Add**.

Note:

You can also set global default image-inspector settings for all OpenShift providers in the advanced settings menu by editing the values under **`ems_kubernetes`**, instead of setting this for each provider.

For example:

```
:image_inspector_registry: registry.access.redhat.com
:image_inspector_repository: openshift3/image-inspector
```

Container Image Scanning

Configuring Image Scanning

Infrastructure Automation manages vulnerability scanning of container images. When an OpenShift provider is added, OpenShift images from the internal registry are discovered. To enable image scanning, perform the following configuration steps:

1. Browse to menu: **Compute > Containers > Providers**.
2. Select the checkboxes of the OpenShift providers on which to enable scanning.
3. From the **Policy** pull-down menu, click **Manage Policies**.
4. Select the **OpenSCAP profile** checkbox.
5. Click **Save**.

This action will trigger a SmartState analysis, or scan, of all images referenced by the OpenShift provider. The initial scan may take several hours to complete, depending on the number and size of images. The scan occurs in the OpenShift provider, which Infrastructure Automation receives and records in the database. OpenShift limits the number of scanning pods; only three images can be scanned simultaneously.

Scheduling A Recurring Scan

Software vulnerability databases are updated frequently. To apply these updates, a rescan is required. To schedule a recurring scan of container images:

:schedule openscap scan

1. Click **Configuration**.

2. From **Settings > Zones** in the left pane of the appliance, select **Schedules**.
3. From the drop-down menu, click **Configuration > Add a new Schedule**.
4. Type an arbitrary **Name**.
5. Type an arbitrary **Description**.
6. Ensure the **Active** checkbox is selected.
7. In **Action**, select **Container Image Analysis**.
8. In **Filter**, select **All Container Images for Containers Provider**, **OpenShift**.
9. In **Run**, set the schedule as desired.
10. Set the **Time Zone**, **Starting Date**, and **Starting Time**.
11. Click **Add**.

Working with Images

Viewing Results

Image scanning results are displayed in each image summary page.

1. Select menu: **Compute > Containers > Container Images**.
2. Click the desired image.

For an OpenSCAP HTML report, locate the **Configuration** section and select **OpenSCAP HTML**.

:container configuration

For compliance and scanning history information, locate the **Compliance** section and note the **Status** field or select **Available** from the **History** field.

:container scan history

Manual Scanning

SmartState analysis scanning may be initiated manually for images. From an image summary page, select **Configuration > Perform SmartState Analysis**. Refreshing the image page will reflect the latest scan results and compliance history.

Evaluating Compliance

If the image scan policy has been updated since the last scan, compliance conditions may be re-evaluated. From an image summary page, select **Policy > Check Compliance of Last Known Configuration**. Refreshing the image page will reflect the latest compliance history.

Generating a Report on Images

You can output the results of an OpenSCAP scan of images to a report for an overview of the security risk level of images. The **Images by Failed OpenSCAP Rule Results are included with Infrastructure Automation and shows whether the image passed or failed OpenSCAP policy criteria, and the security risk.

Note: You can also create a copy of this report and edit it to contain additional information, such as the project name where the image is used to produce more useful results. For more information, see Editing a Report and Reportable Fields in Infrastructure Automation in *Monitoring, Alerts, and Reporting* for instructions on customizing reports.

Steps to create a report to show image compliance:

1. Browse to menu: **Overview > Reports**.
2. Click the menu: **Reports > All Reports** accordion.
3. Browse to menu: **Configuration Management > Containers > Images by Failed OpenSCAP Rule Results** to run a report that shows which images failed the OpenSCAP compliance.
4. Click **Queue**.
5. The report generation is placed in the queue and its status shows in the reports page.

		Queued At	Run At	Source	Username	Group	Status
☐	▶	03/12/18 04:31:57 UTC		Requested by user	ocpadmin	EvmGroup-container_administrator_updated	Queued
☐	→	03/07/18 08:43:25 UTC	03/07/18 08:44:00 UTC	Requested by user	ocpadmin	EvmGroup-container_administrator_updated	Complete

6. Click **(Refresh this page)** to update the status.
7. Browse to the **Saved Reports** accordion, and click the report when it completes.
8. Click the report download buttons for the type of export you want. The report is automatically named with the type of report and date.
   - Click **Download this report in text format** to download as text.
   - Click **Download this report in CSV format** to download as a comma-separated file.
   - Click **Download this report in PDF format** to download as PDF.

OpenSCAP Policy Profile

Infrastructure Automation is pre-configured with a default scanning policy profile. This includes conditions to scan and identify compliance, as well as annotate compliance failure. SmartState analysis is performed when new images are added to OpenShift.

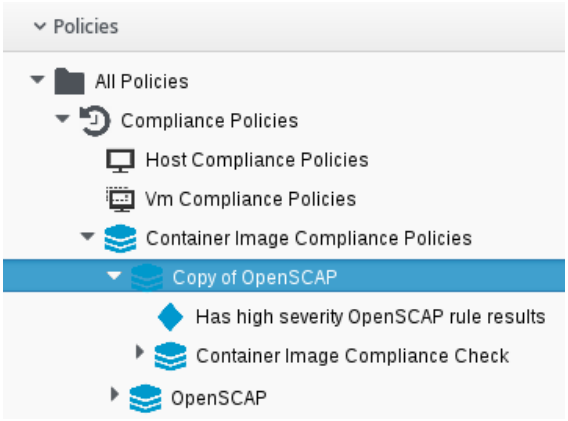
Customizing the Scanning Policy Profile

The built-in OpenSCAP policy profile cannot be edited. However, you can assign *edited* copies of these policies to a new policy profile. This provides you the ability to create a customized version of the built-in OpenSCAP policy profile.

To do so, you must first copy the policy that you want to customize:

1. Browse to menu: **Control > Explorer**.
2. Click the **Policies** accordion, select **Container Image Compliance Policies**, and click **OpenSCAP**.
3. Click **Configuration**, and an option to copy the policy appears; for example, **Copy this Container Image Policy**.
4. Click **OK** to confirm.

The new policy is created with a prefix of **Copy of** in its description, and it can be viewed in the Policies accordion.



You can now edit the copied policy. After editing copied policies, you can add them to a new policy profile. For instructions on how to edit policies, create a new policy profile, and add policies to it, see the Policies and Profiles guide. Once you have a customized policy profile, you can assign it to a containers provider.

Controlling OpenShift Pod Execution

Through the default policy profile, non-compliant images receive the control policy action **Mark as Non-Compliant**. This action annotates the **image** object (not to be confused with the **imagestream** object) with *images.openshift.io/deny-execution=true*. This annotation may be used to prevent nodes from running non-compliant images.

Reference

More information about OpenSCAP, see visit the OpenSCAP web site.

- High Availability Guide
  - Environment Overview
    - Requirements
  - Installing the Appliances
    - Installing the Primary Database-Only Appliance
    - Installing the First Infrastructure Automation Appliance
    - Configuring the Primary Database-Only Appliance
    - Installing the Standby Database-Only Appliance
    - Configuring the Standby Database-Only Appliance
    - Installing Additional Infrastructure Automation Appliances
  - Configuring Database Failover
    - Failover Configuration Settings
    - Configuring the Failover Monitor
    - Testing Database Failover
    - Reintroducing the Failed Node
  - Configuring the HAProxy Load Balancer
    - Verifying the HAProxy Configuration
  - Scaling a Highly Available Infrastructure Automation Environment
  - Updating a Highly Available Infrastructure Automation Environment
  - Updating Hostnames on Database-Only Appliances
    - Preparing to Update Appliance Hostnames
    - Updating the Primary Database-Only Appliance Hostname
    - Updating the Standby Database-Only Appliance Hostname
    - Re-configuring High Availability on Database-Only Appliances
      - Configuring the Primary Database-Only Appliance
      - Configuring the Standby Database-Only Appliance
      - Restarting Services

High Availability Guide

Environment Overview

Learn how to configure and manage database high availability in a Infrastructure Automation environment. This configuration allows for disaster mitigation: a failure in the primary database does not result in downtime, as the standby database takes over the failed database's processes. This is made possible by database replication between two or more database servers. In Infrastructure Automation, these servers are *database-only Infrastructure Automation appliances*, which do not have **`evmserverd`** processes that are enabled. This is configured from the **`appliance_console`** menu at the time of deployment.

Two types of appliances that are used in high availability:

- *Database-only Infrastructure Automation appliances*, which do not have **`evmserverd`** processes that are enabled or a user interface.
- *Non-database Infrastructure Automation appliances*, which are standard appliances that contain a user interface and which have **`evmserverd`** processes that are enabled.

In this configuration, a failover monitor daemon is configured and running on each non-database Infrastructure Automation appliance. The failover monitor watches the **`repmgr`** metadata about the database-only appliances present in the cluster. When the primary database-only appliance goes down, the non-database Infrastructure Automation appliances start polling each of the configured standby database-only appliances to monitor which one comes up as the new primary. The promotion is orchestrated either by **`repmgrd`** on the database-only appliances or is done manually. When the non-database Infrastructure Automation appliances find that a standby has been promoted, Infrastructure Automation reconfigures the setup by writing the new IP address in the **`database.yml`** file to point to the new primary.

Note:

Manual steps are required to reintroduce the failed database node back as the standby server. See Reintroducing the Failed Node.

Note, this procedure also does not provide scalability or a multi-master database setup. While a Infrastructure Automation environment is comprised of an engine tier and a database tier, this configuration affects only the database tier and does not provide load balancing for the appliances.

Requirements

For a high availability Infrastructure Automation environment, you need a virtualization host containing at minimum four virtual machines with Infrastructure Automation installed, consisting of:

- One virtual machine for the primary external database containing a minimum of 4 GB dedicated disk space
- One virtual machine for the standby external database containing a minimum of 4 GB dedicated disk space
- Two virtual machines for the non-database Infrastructure Automation appliances

The database-only appliances must reside on a highly reliable local network in the same location.

Important:

It is essential to use the same Infrastructure Automation appliance template version to install each virtual machine in this environment.

Correct time synchronization is required before installing the cluster. After installing the appliances, configure time synchronization on all appliances by using **`chronyd`**.

Installing the Appliances

This chapter outlines the steps for installing and configuring the Infrastructure Automation components needed for high availability: a database cluster comprised of primary and standby database-only appliances, and two (at minimum) non-database Infrastructure Automation appliances.

Installing the Primary Database-Only Appliance

The primary database-only appliance functions as an external database to the non-database Infrastructure Automation appliances.

1. Deploy a Infrastructure Automation appliance with an extra (and unpartitioned) disk for the database at a size appropriate for your deployment. For recommendations on disk space, see Database Requirements in the *Deployment Planning Guide*.
2. Configure time synchronization on the appliance by editing **`/etc/chrony.conf`** with valid NTP server information.
3. From your host environment, open the appliance and configure the network:
    1. Log in to the appliance and run the **`appliance_console`** command to access the appliance console menu.
    2. Configure networking as desired by selecting the **Set DHCP Network Configuration** or **Set Static Network Configuration** option.
4. Resynchronize time information across the appliances:

```
# systemctl enable chronyd.service
# systemctl start chronyd.service
```

5. In the appliance console, configure the following:
    1. Configure the hostname by selecting **Set Hostname**.
    2. Select **Configure Application**.
    3. Select **Create key** to create the encryption key. You can create a new key, or use an existing key on your system by selecting **Fetch key from remote machine** and following the prompts.
    4. Select **Create Internal Database**.
    5. Select the database disk. Infrastructure Automation then activates the configuration.
    6. For **Should this appliance run as a standalone database server?**, select **`y`**. Selecting this option configures this appliance as a database-only appliance, and therefore the CP4AIOPS:IA application and **`evmserverd`** processes will not run. This is required in highly available database deployments.

        Warning:

        This configuration is not reversible.
    7. Create the database password.

Note:

Do not create a region at this stage in the procedure.

You have now created the empty database.

You can check the configuration on the appliance console details screen. If configured successfully, **Local Database Server** shows as **`running (primary)`**.

Running the **psql vmdb_production** command also provides information about the database.

Installing the First Infrastructure Automation Appliance

Install and configure a Infrastructure Automation appliance to point to the primary database server. You can then create a database region and configure the primary database. This appliance does not serve as a database server.

After installing and configuring an empty database-only appliance in Installing the Primary Database-Only Appliance, the steps in this section create the database schema used by Infrastructure Automation on the primary database-only appliance, and populate the database with the initial data.

Region metadata is required to configure the primary database-only appliance as a primary node in the replication cluster. This must be configured from the Infrastructure Automation appliance before the primary and secondary database-only appliances can be configured.

1. Deploy a Infrastructure Automation appliance. There is no requirement for an extra disk on this appliance.
2. Configure time synchronization on the appliance by editing **/etc/chrony.conf** with valid NTP server information.
3. From your host environment, open the appliance and configure the network:
    1. Log in to the appliance and run the **appliance_console** command to access the appliance console menu.
    2. Configure networking as desired by selecting the **Set DHCP Network Configuration** or **Set Static Network Configuration** option.
4. Re-synchronize time information across the appliances:

```
# systemctl enable chronyd.service
# systemctl start chronyd.service
```

5. In the appliance console, configure the following:
    1. Configure the hostname by selecting **Set Hostname**.
    2. Select **Configure Application**.
    3. Configure this appliance to use the encryption key from the primary database-only appliance:
        1. For **Encryption Key**, select **Fetch key from remote machine**.
        2. Enter the hostname for the primary database-only appliance you previously configured containing the encryption key.
        3. Enter the primary database-only appliance's username.
        4. Enter the primary database-only appliance's password.
        5. Enter the path of the remote encryption key. (For example, **/var/www/miq/vmdb/certs/v2_key**.)

           All appliances in the same region must use the same v2 key.
    4. Configure the database:
        1. Select **Create Region in External Database**, since the database is external to the appliances.

           Creating a database region will destroy any existing data and cannot be undone.
        2. Assign a unique database region number.
        3. Enter the port number.
        4. For **Are you sure you want to continue?** Select **y**.
    5. Enter the primary database-only appliance's name and access details:
        1. Enter the hostname for the primary database-only appliance.
        2. Enter a name to identify the database.
        3. Enter the primary database-only appliance's username.
        4. Enter a password for the database and confirm the password.

This initializes the database, which takes a few minutes.

You can check the configuration on the appliance console details screen. When configured successfully, **CP4AIOPS:IA Server** will show as **running**, and **CP4AIOPS:IA Database** will show the hostname of the primary database-only appliance.

Configuring the Primary Database-Only Appliance

On the primary database-only appliance you created in Installing the Primary Database-Only Appliance, initialize the nodes in the database cluster to configure the database replication. Run these steps from the appliance console:

1. In the appliance console menu, select **Configure Database Replication**.
2. Select **Configure Server as Primary**.
3. Set a unique identifier number for the server and enter the database name and credentials:
    1. Select a number to uniquely identify the node in the replication cluster.
    2. Enter the name of the database you configured previously.
    3. Enter the cluster database username.
    4. Enter the cluster database password and confirm the password.
    5. Enter the primary database-only appliance hostname or IP address.

        Note:

        The hostname must be visible to all appliances that communicate with this database, including the non-database Infrastructure Automation appliances and any global region databases.

    6. Confirm that the replication server configuration details are correct, and select **`y`** to apply the configuration.

This configures database replication in the cluster.

Installing the Standby Database-Only Appliance

The standby database-only appliance is a copy of the primary database-only appliance and takes over the role of primary database in case of failure.

Follow these steps to create a new standby appliance, or to add another standby appliance to the cluster.

1. Deploy a Infrastructure Automation appliance with an extra (and unpartitioned) disk for the database that is the same size as the primary database-only appliance, as it will contain the same data. For recommendations on disk space, see Database Requirements in the *Deployment Planning Guide*.
2. Configure time synchronization on the appliance by editing **`/etc/chrony.conf`** with valid NTP server information.
3. From your host environment, open the appliance and configure the network:
    1. Log in to the appliance and run the **`appliance_console`** command to access the appliance console menu.
    2. Configure networking as desired by selecting the **Set DHCP Network Configuration** or **Set Static Network Configuration** option.
4. Re-synchronize time information across the appliances:

    ```
    # systemctl enable chronyd.service
    # systemctl start chronyd.service
    ```

5. In the appliance console, configure the hostname by selecting **Set Hostname**.

You can now configure this appliance as a standby database-only appliance in the cluster.

Configuring the Standby Database-Only Appliance

The steps to configure the standby database-only appliance are similar to that of the primary database-only appliance, in that they prepare the appliance to be database-only, but as the standby.

On the standby database-only appliance, configure the following from the appliance console:

1. In the appliance console menu, select **Configure Database Replication**.
2. Select **Configure Server as Standby**.
3. Select the database disk. Infrastructure Automation then activates the configuration.
4. Set a unique identifier number for the standby server and enter the database name and credentials:
    1. Select a number to uniquely identify the node in the replication cluster.
    2. Enter the cluster database name.
    3. Enter the cluster database username.

4. Enter and confirm the cluster database password.
5. Enter the primary database-only appliance hostname or IP address.
6. Enter the standby database-only appliance hostname or IP address.

   Note:

   The hostname must be visible to all appliances that communicate with this database, including the engine appliances and any global region databases.
7. Select **`y`** to configure the replication manager for automatic failover.
8. Confirm that the replication standby server configuration details are correct, and select **`y`** to apply the configuration.

The standby server will then run an initial synchronization with the primary database, and start locally in standby mode. This takes a few minutes.

Verify the configuration on the appliance console details screen for the standby server. When configured successfully, **Local Database Server** shows as **`running (standby)`**.

Installing Additional Infrastructure Automation Appliances

Install a second virtual machine with a Infrastructure Automation appliance and any additional appliances in the region using the following steps:

1. Deploy a Infrastructure Automation appliance. There is no requirement for an extra disk on this appliance.
2. Configure time synchronization on the appliance by editing **`/etc/chrony.conf`** with valid NTP server information.
3. From your host environment, open the appliance and configure the network:
   1. Log in to the appliance and run the **`appliance_console`** command to access the appliance console menu.
   2. Configure networking as desired by selecting the **Set DHCP Network Configuration** or **Set Static Network Configuration** option.
4. Re-synchronize time information across the appliances:

   ```
   # systemctl enable chronyd.service
   # systemctl start chronyd.service
   ```
5. In the appliance console, configure the following:
   1. Configure the hostname by selecting **Set Hostname**.
   2. Select **Configure Application**.
   3. Configure this appliance to use the encryption key from the primary database-only appliance:
      1. For **Encryption Key**, select **Fetch key from remote machine**.
      2. Enter the hostname for the primary database-only appliance you previously configured containing the encryption key.
      3. Enter the port number.
      4. Enter the primary database-only appliance's username.
      5. Enter the primary database-only appliance's password.
      6. Enter the path of the remote encryption key. (For example, **`/var/www/miq/vmdb/certs/v2_key`**.)
      7. Select **Join Region in External Database** from the appliance console menu.
   4. Enter the primary database-only appliance's name and access details:
      1. Enter the hostname for the primary database-only appliance.
      2. Enter a name to identify the database.
      3. Enter the primary database-only appliance's username.
      4. Enter a password for the database and confirm the password.

This configuration takes a few minutes to process.

You can check the configuration on the appliance console details screen. When configured successfully, **CP4AIOPS:IA Server** will show as **`running`**, and **CP4AIOPS:IA Database** will show the hostname of the primary database-only appliance.

Configuring Database Failover

The failover monitor daemon must run on all of the non-database Infrastructure Automation appliances to check for failures. In case of a database failure, it modifies the database configuration accordingly.

This configuration is crucial for high availability to work in your environment. If the database failover monitor is not configured, non-database Infrastructure Automation appliances will not react to the failover event and will not be reconfigured against the new primary database host.

Failover Configuration Settings

Configuration settings are located in **`/var/www/miq/vmdb/config/ha_admin.yml`**.

Default settings:

Parameter	Default Value	Description
`failover_attempts`	10	The number of times the **`failover_monitor`** will attempt to failover after discovering the connection to the current database is no longer active.
`db_check_frequency`	120s	The sleep interval in the monitor loop, it defines how frequently the connection to the database is checked.
`failover_check_frequency`	60s	The sleep interval between attempting to failover additional times.

Note:

`failover_attempts` and **`failover_check_frequency`** determine the time limit for the cluster to elect a new primary. To produce quicker failovers, reduce the value for **`failover_check_frequency`** and increase the value for **`failover_attempts`**.

Configuring the Failover Monitor

Configure the failover monitor only on the non-database Infrastructure Automation appliances with the following steps:

1. In the appliance console menu, select **Configure Application Database Failover Monitor**.
2. Select **Start Database Failover Monitor**.

Testing Database Failover

Test that failover is working correctly between your databases with the following steps:

1. Simulate a failure by stopping the database on the primary server:

   ```
   # systemctl stop postgresql.service
   ```

2. To check the status of the database, run:

   ```
   # systemctl status postgresql.service
   ```

   Note:

   You can check the status of the simulated failure by viewing the evm log on the engine appliances with **`journalctl -t evm`**.

3. Check the appliance console summary screen for the primary database. If configured correctly, the **CP4AIOPS:IA Database** value in the appliance console summary should have switched from the hostname of the old primary database to the hostname of the new primary on all Infrastructure Automation appliances.

Upon database server failover, the standby server becomes the primary. However, the failed node cannot switch to standby automatically and must be manually configured. Data replication from the new primary to the failed and recovered node does not happen by default, so the failed node must be reintroduced into the configuration.

Reintroducing the Failed Node

Manual steps are required to reintroduce the failed primary database node back into the cluster as a standby. This allows for greater control over the configuration, and to diagnose the failure.

To reintroduce the failed node, reinitialize the standby database. On the standby database-only appliance, configure the following:

1. In the appliance console menu, select **Configure Database Replication**.
2. Select **Configure Server as Standby**.
3. Select **`y`** to remove all previous data from the server and configure it as a new standby database.
4. Set a unique identifier number for the standby server and enter the database name and credentials:

1. Select a number to uniquely identify the node in the replication cluster. This number can be the same as the node's original identification number.
2. Enter the cluster database name.
3. Enter the cluster database username.
4. Enter the cluster database password.
5. Enter the new primary database-only appliance hostname or IP address.
6. Enter the new standby database-only appliance hostname or IP address.

   Note:

   The hostname must be visible to all appliances that communicate with this database, including the engine appliances and any global region databases.

7. Select **`y`** to configure the replication manager for automatic failover.

   Note:

   If re-using the node's identification number, select **`y`** to overwrite the existing node ID (this cannot be undone). Additionally, select **`y`** to overwrite and reconfigure the replication settings in **`/etc/repmgr/*/repmgr.conf`** when prompted.

8. Confirm that the replication standby server configuration details are correct, and select **`y`** to apply the configuration.

The standby server will then run an initial synchronization with the primary database, and start locally in standby mode.

Verify the configuration on the appliance console details screen for the standby server. When configured successfully, **Local Database Server** shows as **`running (standby)`**.

Your Infrastructure Automation environment is now re-configured for high availability.

Configuring the HAProxy Load Balancer

After configuring the appliances as described in Installing the Appliances, configure a load balancer to direct traffic to the non-database Infrastructure Automation appliances.

The following steps highlight the configuration requirements for the load balancer, which in this case is HAProxy. The load balancer is assigned a virtual IP address for the Infrastructure Automation user interface and is pointed to one of the many non-database Infrastructure Automation appliances behind the load balancer in a round robin fashion.

Additionally, to avoid the HAProxy server being a single point of failure, two redundant HAProxy servers are configured in active-passive mode. The failover is orchestrated by using the **`keepalived`** daemon. **`keepalived`** monitors the health of the active load balancer and in case of a failure, the virtual IP is failed over to the passive load balancer, which then becomes active. The virtual IP is configured by `keepalived`.

The virtual IP is the single IP address that is used as the point of access to the Infrastructure Automation appliance user interfaces and is configured in the HAProxy configuration along with a load balancer profile. When an end user accesses the virtual IP, it directs traffic to the appropriate Infrastructure Automation appliance based on the configured HAProxy policy.

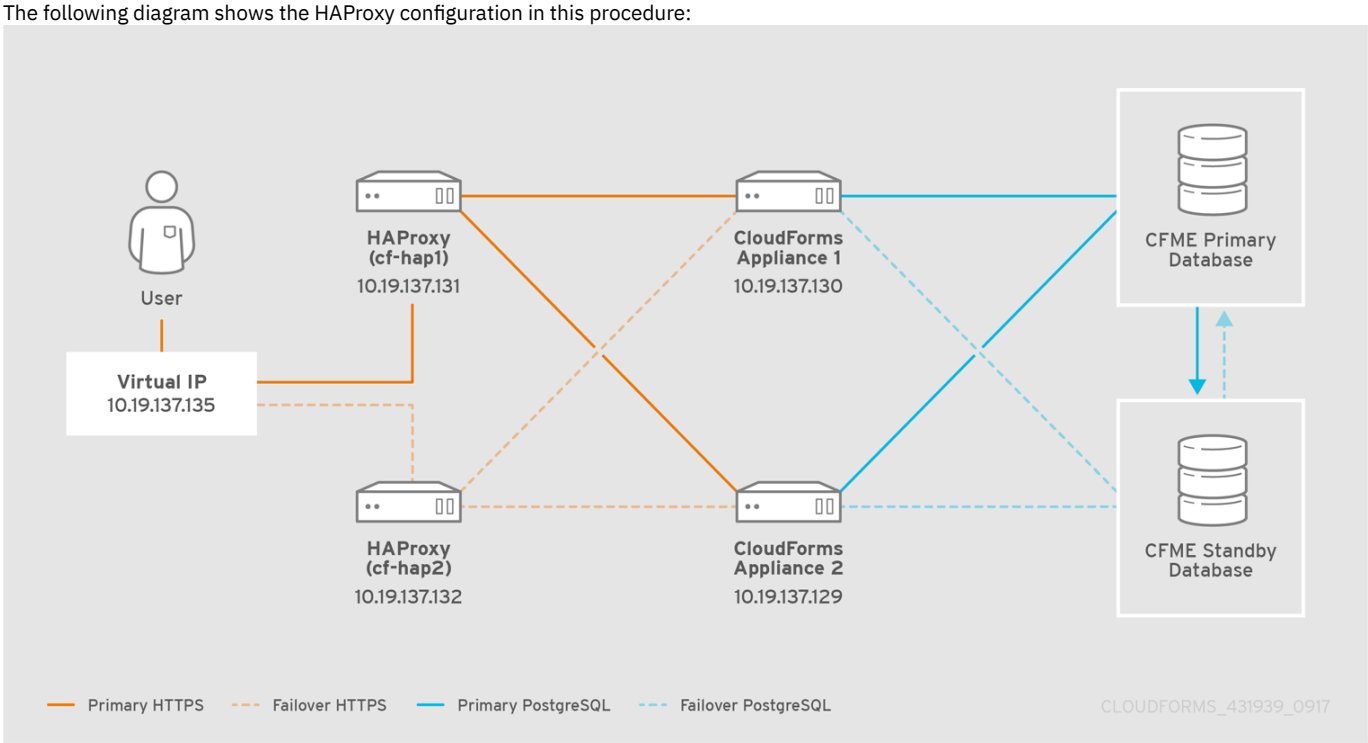
Note:

Additional configuration is required to run HAProxy on Red Hat OpenStack Platform. See the OpenStack Networking Guide for more information.

This configuration uses two HAProxy servers and a virtual IP (configured by **`keepalived`**). The following example procedure uses the following IP addresses and names; substitute values for your environment as needed:

- HAProxy1: 10.19.137.131 (cf-hap1.example.com)
- HAProxy2: 10.19.137.132 (cf-hap2.example.com)
- Virtual IP (to be configured by **`keepalived`**): 10.19.137.135 (cf-ha.example.com)
- CP4AIOPS:IA Appliance 1: 10.19.137.130 (cfme1.example.com)
- CP4AIOPS:IA Appliance 2: 10.19.137.129 (cfme2.example.com)

The following diagram shows the HAProxy configuration in this procedure:



To configure HAProxy load balancing:

1. Install two servers (virtual or physical) running Red Hat Enterprise Linux 7.2 or above, to be used as the HAProxy servers.
2. Configure subscriptions on both HAProxy servers (**cf-hap1** and **cf-hap2**) so that the **rhel-7-server-rpms** repository is enabled:

```
[root@cf-hap1 ~]# subscription-manager repos --list-enabled
+----------------------------------------------+
Available Repositories in /etc/yum.repos.d/redhat.repo
+----------------------------------------------+
Repo Name:   Red Hat Enterprise Linux Server 7 Server (RPMs)

Repo ID:     rhel-7-server-rpms

Repo URL: https://cdn.redhat.com/content/dist/rhel/server/7/$release/$basearch/os
Enabled: 1

[root@cf-hap2 ~]# subscription-manager repos --list-enabled
+----------------------------------------------+
Available Repositories in /etc/yum.repos.d/redhat.repo
+----------------------------------------------+
Repo Name:   Red Hat Enterprise Linux Server 7 Server (RPMs)

Repo ID:     rhel-7-server-rpms

Repo URL: https://cdn.redhat.com/content/dist/rhel/server/7/$release/$basearch/os
Enabled: 1
```

3. Configure the firewall on both HAProxy servers.
    1. On the **cf-hap1** server, run the following:

        Note:

        keepalived uses VRRP (Virtual Redundancy Router Protocol) to monitor the servers and determine which node is the master and which node is the backup. VRRP communication between routers uses multicast IPv4 address 224.0.0.18 and IP protocol number 112.

```
[root@cf-hap1 ~]# firewall-cmd --permanent  --add-port=80/tcp --add-port=443/tcp --add-
port=8443/tcp && firewall-cmd --reload

[root@cf-hap1 ~]# firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 \ --in-interface
eth0 --destination 224.0.0.18 --protocol vrrp -j ACCEPT

[root@cf-hap1 ~]#firewall-cmd --direct --permanent --add-rule ipv4 filter OUTPUT 0 \ --out-
interface eth0 --destination 224.0.0.18 --protocol vrrp -j ACCEPT

[root@cf-hap1 ~]# firewall-cmd --reload
```

2. On the **cf-hap2** server, repeat the same commands by running the following:

```
[root@cf-hap2 ~]# firewall-cmd --permanent  --add-port=80/tcp --add-port=443/tcp --add-
port=8443/tcp && firewall-cmd --reload

[root@cf-hap2 ~]# firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 \ --in-interface
eth0 --destination 224.0.0.18 --protocol vrrp -j ACCEPT

[root@cf-hap2 ~]# firewall-cmd --direct --permanent --add-rule ipv4 filter OUTPUT 0 \ --out-
interface eth0 --destination 224.0.0.18 --protocol vrrp -j ACCEPT

[root@cf-hap2 ~]# firewall-cmd --reload
```

4. Install and configure **keepalived** on both servers.

1. On the **cf-hap1** server, run the following:

```
[root@cf-hap1 ~]# yum install keepalived -y

[root@cf-hap1 ~]# cat /etc/keepalived/keepalived.conf
vrrp_script chk_haproxy {
script "killall -0 haproxy" # check the haproxy process
interval 2 # every 2 seconds
weight 2 # add 2 points if OK
}
vrrp_instance VI_1 {
interface eth0              # interface to monitor
state MASTER              # MASTER on haproxy1, BACKUP on haproxy2
virtual_router_id 51
priority 101              # 101 on haproxy1, 100 on haproxy2
virtual_ipaddress {
10.19.137.135/21 # virtual ip address
}
track_script {
chk_haproxy
}
}
```

2. On the **cf-hap2** server, run the following:

```
[root@cf-hap2 ~]# yum install keepalived -y

[root@cf-hap2 ~]# cat /etc/keepalived/keepalived.conf
vrrp_script chk_haproxy {
script "killall -0 haproxy" # check the haproxy process
interval 2 # every 2 seconds
weight 2 # add 2 points if OK
}
vrrp_instance VI_1 {
interface eth0              # interface to monitor
state BACKUP              # MASTER on haproxy1, BACKUP on haproxy2
virtual_router_id 51
priority 100              # 101 on haproxy1, 100 on haproxy2
virtual_ipaddress {
10.19.137.135/21 # virtual ip address
}
track_script {
chk_haproxy
}
}
```

3. On both servers, configure IP forwarding and non-local binding by appending the following to the **sysctl.conf** file. In order for the **keepalived** service to forward network packets properly to the real servers, each router node must have IP forwarding turned on in the kernel. On the **cf-hap1** server, run the following:

```
[root@cf-hap1 ~]# cat /etc/sysctl.conf
# System default settings live in /usr/lib/sysctl.d/00-system.conf.
# To override those settings, enter new settings here, or in an /etc/sysctl.d/<name>.conf file
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1
```

4. On the **cf-hap2** server, run the following:

```
[root@cf-hap2 ~]# cat /etc/sysctl.conf
# System default settings live in /usr/lib/sysctl.d/00-system.conf.
# To override those settings, enter new settings here, or in an /etc/sysctl.d/<name>.conf file
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
```

```
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1
```

5. Verify that the **sysctl.conf** settings were saved on each server:

```
[root@cf-hap1 ~]# sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1

[root@cf-hap2 ~]# sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.ip_nonlocal_bind = 1
```

5. Install HAProxy on both servers:

```
[root@cf-hap1 ~]# yum install haproxy -y

[root@cf-hap2 ~]# yum install haproxy -y
```

6. Configure the appropriate IPs for load balancing on the **cf-hap1** server as follows:

```
[root@cf-hap1 ~]# cat /etc/haproxy/haproxy.cfg
global
    log                 127.0.0.1 local0
    chroot              /var/lib/haproxy
    pidfile             /var/run/haproxy.pid
    maxconn         4000
    user                haproxy
    group               haproxy
    daemon
defaults
    mode                        http
    log                         global
    option                      httplog
    option                      dontlognull
    option             http-server-close
    option     forwardfor       except 127.0.0.0/8
    option                      redispatch
    retries                     3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client              1m
    timeout server          1m
    timeout http-keep-alive     10s
    timeout check           10s
# Infrastructure Automation Management UI URL
listen apache
  bind 10.19.137.135:80
  mode tcp
  balance source
  server cfme1 10.19.137.130:80 check inter 1s
  server cfme2 10.19.137.129:80  check inter 1s
#
listen apache-443
  bind 10.19.137.135:443
  mode tcp
  balance source
  server cfme1 10.19.137.130:443 check inter 1s
  server cfme2 10.19.137.129:443  check inter 1s
#
listen apache-8443
  bind 10.19.137.135:8443
  mode tcp
  balance source
  server cfme1 10.19.137.130:8443 check inter 1s
  server cfme2 10.19.137.129:8443  check inter 1s
```

Note:

- The virtual IP in this configuration is 10.19.137.135 (cf-haproxy.example.com).
- The IP of CP4AIOPS:IA Appliance 1 is 10.19.137.130 (cfme1.example.com).
- The IP of CP4AIOPS:IA Appliance 2 is 10.19.137.129 (cfme2.example.com).

7. Configure the appropriate IPs for load balancing on the **cf-hap2** server as well:

```
[root@cf-hap2 ~]# cat /etc/haproxy/haproxy.cfg
global
    log                 127.0.0.1 local0
```

```
    chroot              /var/lib/haproxy
    pidfile             /var/run/haproxy.pid
    maxconn         4000
    user                haproxy
    group               haproxy
    daemon
defaults
    mode                        http
    log                         global
    option                      httplog
    option                      dontlognull
    option             http-server-close
    option     forwardfor       except 127.0.0.0/8
    option                      redispatch
    retries                     3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client              1m
    timeout server          1m
    timeout http-keep-alive     10s
    timeout check           10s
# Infrastructure Automation Management UI URL
listen apache
  bind 10.19.137.135:80
  mode tcp
  balance source
  server cfme1 10.19.137.130:80 check inter 1s
  server cfme2 10.19.137.129:80  check inter 1s
#
listen apache-443
  bind 10.19.137.135:443
  mode tcp
  balance source
  server cfme1 10.19.137.130:443 check inter 1s
  server cfme2 10.19.137.129:443  check inter 1s
#
listen apache-8443
  bind 10.19.137.135:8443
  mode tcp
  balance source
  server cfme1 10.19.137.130:8443 check inter 1s
  server cfme2 10.19.137.129:8443  check inter 1s
```

8. On each server, start the **keepalived** and **haproxy** services:

```
[root@cf-hap1~]# systemctl enable keepalived
[root@cf-hap1~]# systemctl start keepalived
[root@cf-hap1~]# systemctl enable haproxy
[root@cf-hap1~]# systemctl start haproxy

[root@cf-hap2~]# systemctl enable keepalived
[root@cf-hap2~]# systemctl start keepalived
[root@cf-hap2~]# systemctl enable haproxy
[root@cf-hap2~]# systemctl start haproxy
```

Verifying the HAProxy Configuration

Verify the HAProxy configuration by inspecting the following:

On the master node (**cf-hap1**):

```
[root@cf-hap1 ~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:01:a4:ac:32:4e brd ff:ff:ff:ff:ff:ff
    inet 10.19.137.131/21 brd 10.19.143.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 10.19.137.135/21 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 2620:52:0:1388:201:a4ff:feac:324e/64 scope global mngtmpaddr dynamic
       valid_lft 2591800sec preferred_lft 604600sec
    inet6 fe80::201:a4ff:feac:324e/64 scope link
       valid_lft forever preferred_lft forever
```

On the backup node (**cf-hap2**):

```
[root@cf-hap2 ~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:01:a4:ac:33:a6 brd ff:ff:ff:ff:ff:ff
    inet 10.19.137.132/21 brd 10.19.143.255 scope global eth0
       valid_lft forever preferred_lft forever
```

```
    inet6 2620:52:0:1388:201:a4ff:feac:33a6/64 scope global noprefixroute dynamic
       valid_lft 2591982sec preferred_lft 604782sec
    inet6 fe80::201:a4ff:feac:33a6/64 scope link
       valid_lft forever preferred_lft forever
```

Notice the virtual IP 10.19.137.135 has been started by **keepalived** (VRRP).

Simulate a failure on the master node:

```
[root@cf-hap1 ~]# systemctl stop keepalived
```

Notice the virtual IP failover on the master node (**cf-hap1**):

```
[root@cf-hap1 ~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:01:a4:ac:32:4e brd ff:ff:ff:ff:ff:ff
    inet 10.19.137.131/21 brd 10.19.143.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 2620:52:0:1388:201:a4ff:feac:324e/64 scope global mngtmpaddr dynamic
       valid_lft 2591800sec preferred_lft 604600sec
    inet6 fe80::201:a4ff:feac:324e/64 scope link
       valid_lft forever preferred_lft forever
```

The backup node (**cf-hap2**) shows the following:

```
[root@cf-hap2 ~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:01:a4:ac:33:a6 brd ff:ff:ff:ff:ff:ff
    inet 10.19.137.132/21 brd 10.19.143.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 10.19.137.135/21 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 2620:52:0:1388:201:a4ff:feac:33a6/64 scope global noprefixroute dynamic
       valid_lft 2591982sec preferred_lft 604782sec
    inet6 fe80::201:a4ff:feac:33a6/64 scope link
       valid_lft forever preferred_lft forever
```

Your environment is now configured for high availability.

The following additional configuration in the Infrastructure Automation user interface worker appliances and the load balancer are recommended for improved performance in worker appliances: - For each Infrastructure Automation appliance behind the load balancer, change the `session_store` setting to `sql` in the appliance's advanced settings. - Configure sticky sessions in the load balancer. - Configure the load balancer to test for appliance connectivity using the `https://appliance_name/ping` URL.

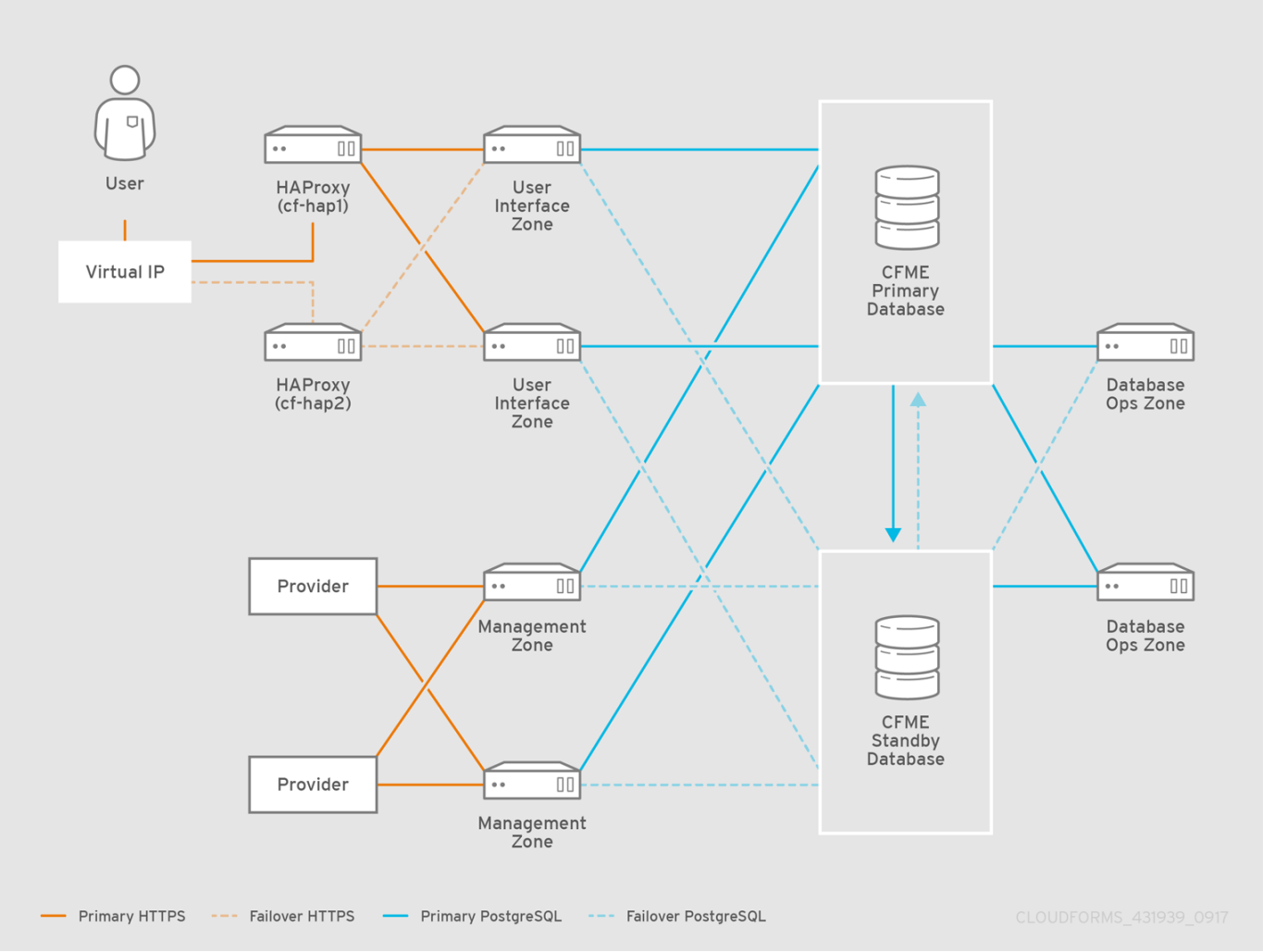
Scaling a Highly Available Infrastructure Automation Environment

After creating high availability for the database tier and the user interface tier, the rest of the infrastructure should be sized appropriately for the roles and the environments that they manage. These roles and tiers use built-in high availability mechanisms like primary, secondary, and tertiary failover.

You can configure additional worker appliances as needed using the steps in Installing Additional Appliances, and then assign zones and server roles. The non-database Infrastructure Automation appliances and roles can be configured in any order.

The following diagram shows an example of a highly available database configuration that contains worker appliances, providers, and the HAProxy load balancer configured in Configuring the HAProxy Load Balancer.

The worker appliances in the diagram are labeled by server role (**User Interface**, **Management**, and **Database Ops**) and corresponding zone to show how a highly available environment might be scaled with server roles and zones.



See Regions and Servers in *General Configuration* for more information on configuring servers and roles.

Updating a Highly Available Infrastructure Automation Environment

Applying software package minor updates (referred to as *errata*) to appliances in a high availability environment must be performed in a specific order to avoid migrating your databases to the next major Infrastructure Automation version.

Prerequisites.

Ensure each appliance is registered to Red Hat Subscription Manager and subscribed to the update channels required by Infrastructure Automation in order to access updates.

To verify if your appliance is registered and subscribed to the correct update channels, run:

```
# yum repolist
```

Appliances must be subscribed to the following channels:

- {product-repo_cfme}
- {product-repo_ansible}
- {product-repo_extras}
- {product-repo_optional}
- {product-repo_server}

Updating the Appliances.

Follow this procedure to update appliances in your environment without migrating the database to the next major version of Infrastructure Automation. Note the appliance to perform each step on: some steps are to be performed only on the database-only appliances, and other steps only on the non-database Infrastructure Automation appliances, while some steps apply to all appliances.

1. Power off the non-database Infrastructure Automation appliances.
2. Power off the database-only appliances.
3. Back up each appliance:
    1. Back up the database of your appliance. Take a snapshot if possible.
    2. Back up the following files for disaster recovery, noting which appliance each comes from:
        - **/var/www/miq/vmdb/GUID**
        - **/var/www/miq/vmdb/REGION**
    3. Note the hostnames and IP addresses of each appliance. This information is available on the summary screen of the appliance console.
4. Start each database-only appliance.
5. Start each non-database Infrastructure Automation appliance again, and stop **evmserverd** on each just after boot:

    ```
    # systemctl stop evmserverd
    ```

6. Apply updates by running the following on each appliance:

    ```
    # yum update
    ```

7. On one of the non-database Infrastructure Automation appliances, apply any database schema updates included in the errata, and reset the Red Hat and ManageIQ automation domains:

    ```
    # vmdb
    # rake db:migrate
    # rake evm:automate:reset
    ```

8. Power off the non-database Infrastructure Automation appliances.
9. Reboot the database-only appliances.
10. Wait five minutes, then start the non-database Infrastructure Automation appliances again.

The appliances in your high availability environment are now up to date.

Updating Hostnames on Database-Only Appliances

When updating the hostnames of database-only appliances in a cluster, you must also re-configure high availability on the primary and standby database-only appliances.

Preparing to Update Appliance Hostnames

Before updating the hostnames of your appliances, complete the following:

1. Note the hostnames of the active database (primary) appliance and the standby database appliances in the appliance console.

    You can verify this from the appliance summary screen of a database-only appliance, where the status for **Local Database Server** shows *primary* or *standby*, for example:

    ```
    ...
    Local Database Server:   running (primary)
    ...
    ```

2. On the non-database Infrastructure Automation appliances, stop the failover monitor:
    1. In the appliance console menu, select **Configure Application Database Failover Monitor**.
    2. Select **Stop Database Failover Monitor**.
3. On the *standby* database-only appliances, stop the **postgresql** database service:

    ```
    # systemctl stop postgresql.service
    ```

4. Stop **evmserverd** on each non-database Infrastructure Automation appliance:

    ```
    # systemctl stop evmserverd
    ```

Updating the Primary Database-Only Appliance Hostname

Run the following steps on your primary database-only appliance:

1. In the appliance console menu, configure the new hostname or IP address from the **Configure Network** option.
2. Restart the appliance.
3. Update the **`host`** key to the primary database appliance's new hostname in **`/var/www/miq/vmdb/config/database.yml`**.

Updating the Standby Database-Only Appliance Hostname

Run the following steps on your standby database-only appliances:

1. In the appliance console menu, configure the new hostname or IP address from the **Configure Network** option.
2. Restart the appliance.
3. Update the **`host`** key to the standby database appliance's new hostname in **`/var/www/miq/vmdb/config/database.yml`**.

Repeat these steps on any additional standby database-only appliances.

Your primary and standby appliances must be reachable to each other by their hostnames. If all appliances are on the same network and are resolvable, no additional updates are needed. If your appliances exist on different networks, edit the `/etc/hosts` file on each database appliance to include entries for the IP address and hostname of each other database appliance in the cluster.
Proceed to re-configure high availability on the primary and standby database-only appliances.

Re-configuring High Availability on Database-Only Appliances

Re-configure replication on the database-only appliances, and restart services on your cluster.

Configuring the Primary Database-Only Appliance

On the primary database-only appliance, initialize the nodes in the database cluster to re-configure the database replication:

1. In the appliance console menu, select **Configure Database Replication**.
2. Select **Configure Server as Primary**.
3. Set a unique identifier number for the server and enter the database name and credentials:
    1. Select a number to uniquely identify the node in the replication cluster.
    2. Enter the cluster database name.
    3. Enter the cluster database username.
    4. Enter the cluster database password and confirm the password.
    5. Enter the new primary database-only appliance hostname or IP address.

       Note:

       The hostname or IP address must be visible to all appliances that communicate with this database, including the non-database Infrastructure Automation appliances and any global region databases.
    6. Confirm that the replication server configuration details are correct, and select **`y`** to apply the configuration.

Configuring the Standby Database-Only Appliance

The steps to re-configure the standby database-only appliances are similar to that of the primary database-only appliance, in that they prepare the appliance to be database-only, but as the standby.

On the standby database-only appliances, configure the following:

1. In the appliance console menu, select **Configure Database Replication**.
2. Select **Configure Server as Standby**.
3. Select the database disk. Infrastructure Automation then activates the configuration.
4. Set a unique identifier number for the standby server and enter the database name and credentials:
    1. Select a number to uniquely identify the node in the replication cluster.
    2. Enter the cluster database name.
    3. Enter the cluster database username.
    4. Enter the cluster database password.

5. Enter the new primary database-only appliance hostname or IP address.
6. Enter the new standby database-only appliance hostname or IP address.

   Note:

   The hostname or IP address must be visible to all appliances that communicate with this database, including the engine appliances and any global region databases.
7. Select **`y`** to configure the replication manager for automatic failover.
8. Confirm that the replication standby server configuration details are correct, and select **`y`** to apply the configuration. The standby server will then run an initial synchronization with the primary database, and start locally in standby mode.

5. Verify the configuration on the appliance console details screen for the standby server. When configured successfully, **Local Database Server** shows as **`running (standby)`**.

Repeat these steps on any additional standby database-only appliances.

If you are using non-dedicated database appliances, also stop `evmserverd` on those appliances before changing their hostnames, and reconfigure `database.yml` before restarting.

Restarting Services

1. Start **`evmserverd`** on each non-database Infrastructure Automation appliance:

   ```
   # systemctl start evmserverd
   ```

   After **`evmserverd`** has started successfully, all appliances will be able connect to the database.
2. Restart the failover monitor on the non-database Infrastructure Automation appliances:
   1. In the appliance console menu, select **Configure Application Database Failover Monitor**.
   2. Select **Start Database Failover Monitor**.

Note:

You can view a summary of the updated appliances by running **`repmgr cluster show`** on one of the database appliances.

Your Infrastructure Automation environment is now re-configured for high availability.

- Appliance Hardening Guide
- Introduction to Hardening Infrastructure Automation
  - SSH Security
    - Setting the Root Password on the Appliance
    - Setting SSH Keys on the Appliance
  - Appliance Security
    - Setting the Password for the Administrative User
    - Configuring Host-Based Access Control Rules on your IPA Server
  - Server Security
    - Configuring Firewall Ports
    - Generating SSL Certificates for Your Appliance and Database
      - Creating a Certificate Signing Request
      - Creating a Self-signed Certificate
      - Enabling Your Certificate
    - Creating Custom Encryption Keys
    - Applying SCAP Standards
  - Database Security
    - Restricting Hosts Access to the Database
    - Configuring the Database to use SSL
      - Configure SSL on the database appliance
      - Configure SSL on all connected appliances
      - Hardening TLS Protocol Version

Appliance Hardening Guide

Introduction to Hardening Infrastructure Automation

The Infrastructure Automation appliance is distributed as a virtual machine image, which provides users with a simple installation process. The appliance also contains a set of factory default settings that can expose appliances to vulnerabilities if left unset. This guide provides a set of procedures to enhance security on your Infrastructure Automation appliance. This ensures your appliance has enhanced protection against any unwarranted intrusion.

It is recommended to perform these steps immediately after installing all appliances in your infrastructure.

SSH Security

Setting the Root Password on the Appliance

The Infrastructure Automation appliance is a virtual machine image that runs on a Red Hat Enterprise Linux-based operating system. This means users can access the base operating system through SSH. This is why it is advisable to change the default password. Continuing to use the default password leaves the appliance vulnerable to any user attempting to gain root access.

Changing the **root** password on the appliance 'uses' **the** same process as changing any user password on a Linux-based system.

1. Access your appliance through SSH as the **root** user:

   ```
   [user@localhost ~]$ ssh root@10.1.1.205
   ```

   Substitute **10.1.1.205** with the address of your appliance.

2. Enter the **passwd** command, which changes the password for the current user:

   ```
   [root@ ~]# passwd
   ```

3. Enter and Confirm and new password for the **root** user.

   ```
   Changing password for user root.
   New password: ************
   Confirm password: ************
   ```

4. Log out of the appliance.

The Infrastructure Automation appliance now has a non-default **root** password. This prevents unauthorized access to your appliance through SSH.

Setting SSH Keys on the Appliance

Another recommended practice is to use SSH keys to access the appliance from a single machine. An SSH key provides access from one machine to another through the SSH protocol. The following procedure shows how to create an SSH key on your local machine and add it to the appliance.

1. Check the **.ssh/** directory in your home directory for any existing key pairs:

   ```
   [user@localhost ~]$ ls ~/.ssh/
   ```

   A key pair usually consists of two files. One file is the private key, which stays on your local machine, and the other is the public key, which you copy to another machine. But files are named the same except the public key ends with a **.pub** extension.

   If a key pair already exists, you can use this key pair. Otherwise, use the next few steps to create your own.

2. On your local machine, start the key pair generation process using the **ssh-keygen** command:

   ```
   [user@localhost ~]$ ssh-keygen -t rsa
   ```

3. A prompt asks for the file and location to store these keys:

   ```
   Enter file in which to save the key (/home/user/.ssh/id_rsa):
   ```

   Accept the default path if you do not have a **id_rsa** key pair.

4. Another prompt asks for a passphrase:

   ```
   Enter passphrase (empty for no passphrase):
   ```

   This encrypts the key pair with a password. This protects the key pair if it ever falls into the wrong hands. Alternatively, you can leave the passphrase empty, which provides an automatic login between your local machine and the remote machine.

5. The **ssh-keygen** command generates two files:

   - The private key - the default is */home/user/.ssh/id_rsa*

- The public key - the default is */home/user/.ssh/id_rsa.pub*

  Copy the public key to the appliance using the **ssh-copy-id** command:

  ```
  [user@localhost ~]$ ssh-copy-id ~/.ssh/id_rsa.pub root@10.1.1.205
  ```

  The command copies the public key to the appliance. You might receive a prompt for the password of the root user on the appliance.

6. Test the SSH key authentication:

   ```
   [user@localhost ~]$ ssh root@10.1.1.205
   ```

   This authenticates using the SSH key pair. If you entered a passphrase for the key, the command prompts you for the passphrase.

7. As an additional security measure, edit the **/etc/ssh/sshd_config** on the appliance and modify the following parameter:

   ```
   PermitRootLogin without-password
   ```

   This forces the **root** user account to use certificates instead of passwords for SSH login. This means only your local system can access the appliance.

The appliance now restricts access to only a single machine using the SSH key.

Appliance Security

Setting the Password for the Administrative User

Infrastructure Automation uses a unique **admin** user to control all functions in the web-based user interface. After installing the appliance, change the default password of the **admin** to restrict administrative access to the appliance's UI.

Changing the **admin** password uses the same process as changing any standard user in the appliance.

1. Access the appliance through your web browser and log in.
2. Click (**Configuration**).
3. In the accordion tree on the left, click on **Access Control**, then select the **Administrator** under the **Users** section. This displays the details for the **admin** user.
4. On the details page, select **Configuration > Edit this user** from the toolbar.
5. Enter a new password in the **Change Password / Confirm Password** fields.
6. Click **Save** at the bottom of the page.
7. Log out of the user interface.
8. Test your new password by logging into the user interface. Additionally, test your new password in the appliance console.

The Infrastructure Automation appliance now has a non-default **admin** password. This restricts access to your appliance's administrative functions.

Configuring Host-Based Access Control Rules on your IPA Server

Infrastructure Automation provides support for external authentication using an IPA server. However, there are certain recommendations to enhance security to your appliance, such as creating a specific user group and host group that can access the appliance authentication service.

Run the following steps on your IPA server:

1. Create a user group and restrict access to only the Infrastructure Automation users:

   ```
   [root@ipa ~]# ipa group-add infrastructure_automation_users --desc="infrastructure_automation Users"
   [root@ipa ~]# ipa group-add-member infrastructure_automation_users --users=testuser1,testuser2
   ```

2. Create a host group and restrict access to your appliance hosts:

   ```
   [root@ipa ~]# ipa hostgroup-add infrastructure_automation_hosts --desc "Infrastructure Automation hosts"
   [root@ipa ~]# ipa hostgroup-add-member infrastructure_automation_hosts --
   hosts=appliance1.example.com,appliance2.example.com
   ```

3. Add rules to allow the host group and user group access to the Infrastructure Automation HTTP service:

   ```
   [root@ipa ~]# ipa hbacrule-add infrastructure_automation_access --srchostcat=all
   [root@ipa ~]# ipa hbacrule-add-service infrastructure_automation_access --hbacsvcs httpd-auth
   [root@ipa ~]# ipa hbacrule-add-user infrastructure_automation_access --groups
   infrastructure_automation_users
   ```

```
[root@ipa ~]# ipa hbacrule-add-host infrastructure_automation_access --hostgroups
infrastructure_automation_hosts
```

4. Remove the default rule on your IPA server to allow access to all:

```
[root@ipa ~]# ipa hbacrule-disable allow_all
```

This ensures only users in the **infrastructure_automation_users** group can access the authentication service (**http-auth**) on the appliances in the **infrastructure_automation_hosts** host group.

Server Security

Configuring Firewall Ports

A new appliance starts with a few standard ports open:

- 22 for SSH communication
- 80 for HTTP access to the appliance
- 443 for HTTPS access to the appliance
- 5432 for the appliance database

You might need to restrict or open access to certain services on your appliance in the future. In such situations, use the following method:

- Use **firewalld** to enable a service or port, specifying the zone in use. For example, to open the LDAP port:

```
[root@ ~]# firewall-cmd --zone=manageiq --permanent --add-port=389/tcp
```

The following table lists the appliance's main services and their respective ports.

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
Administrator (Internet Browser)	CFME appliance (User Interface)	HTTPS	443		Access to CFME appliance User Interface
Administrator (Internet Browser)	CFME appliance (User Interface)	HTTP	80		Redirect Web Browser to HTTPS service (443)
Service Catalog or other integration through Web Service	CFME appliance (Web Service)	HTTPS	443		Access to CFME appliance Web Service
CFME appliance	NFS Server	NFS	2049	2049	Embedded NFS VM scanning
CFME appliance (User Interface)	Any Virtual Machine	TCP	903		VM Remote Console (if using MKS plug-in)
CFME appliance (User Interface)	Any Virtual Machine	TCP	5900-5999		VM Remote Console (if using VNC)
CFME appliance (any role)	CFME appliance running the VMDB	PostgreSQL Named Pipes	5432		CFME appliance connectivity to the CFME Database (PostgreSQL)
CFME Subordinate Region VMDB appliance (Database Operations)	CFME Master Region VMDB appliance	PostgreSQL Named Pipes	5432		Regional VMDB node replication up to Master VMDB node (PostgreSQL only)
CFME Subordinate Region VMDB appliance	CFME Master Region VMDB appliance (Web Services and/or User Interface)	PostgreSQL Named Pipes	5432		Subscription validation (PostgreSQL only)
CFME appliance(Authentication through LDAP)	LDAP Server (AD or other)	LDAP	389		LDAP integration

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME appliance (Authentication through LDAPs)	LDAP Server (AD or other)	LDAPs	636		LDAPS integration
SNMP Agent	CFME appliance (Notifier)	SNMP (UDP)		161	SNMP Polling
CFME appliance (Notifier)	SNMP Server	SNMP (TCP)	162		SNMP Trap Send
CFME appliance (Notifier)	Mail server	SMTP	25		SNMP Trap Send
CFME appliance (any role)	NTP Server	NTP		123	Time Source
CFME appliance	CFME SmartProxy installed on VMware ESX Server	HTTPS	1139		Communication with SmartProxy
CFME appliance	DNS Server	UDP		53	DNS Lookups

Ports Used by Infrastructure Automation

The following tables detail the ports used by Infrastructure Automation to communicate with providers.

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME appliance (SmartProxy)	RHEV-M Server	HTTPS	8443		API communications to RHEV-M environment (Inventory, Operations, SmartProxy)
CFME appliance (C\&U)	RHEV-M Server	PostgreSQL	5432		RHEV-M History Database (Database connectivity not enabled by default). See How to access the RHEV-M Postgres DB from a remote machine.
CFME appliance	RHEV-H Hosts or RHEL Hypervisors	SSH	22		SSH connections.
CFME appliance	RHEV-H Hosts or RHEL Hypervisors	DirectLUN			Direct LUN hook must be installed and enabled for embedded VM scanning on FC or iSCSI storage devices. Not a tcp/udp connection.

Red Hat Enterprise Virtualization Ports Used by Infrastructure Automation

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME appliance	RHOS (Keystone)	HTTP REST API	5000		Authentication and Service Entry Point
CFME appliance	RHOS (Nova)	HTTP REST API	8774		Compute Resources
CFME appliance (C\&U)	RHOS (Ceilometer)	HTTP REST API	8777		Metrics for Capacity and Utilization
CFME appliance	RHOS (Glance)	HTTP REST API	9292		Authentication and Service Entry Point
CFME appliance	RHOS (AMQP)	AMQP	5672		Events Integration
CFME appliance	RHOS (Neutron)	HTTP REST API	9696		Networking
CFME appliance	RHOS (Cinder)	HTTP REST API	8776		Block Storage

Red Hat OpenStack Platform Ports Used by Infrastructure Automation

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME Appliance	OpenShift Master Node(s) (or Load Balancer)	HTTPS	8443 or 443		Required for communication to the OpenShift API. Dependent on OpenShift configuration.
CFME Appliance	OpenShift Infrastructure Node(s) (or Load Balancer)	HTTPS	443		Metrics and logging

OpenShift Container Platform Ports Used by Infrastructure Automation

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME appliance(Management System Inventory, Management System Operations, C & U Data Collection, SmartProxy)	vCenter	HTTPS	443		CFME appliance running any of these roles will initiate communication with vCenter on this port
CFME appliance (SmartProxy)	ESX, ESXi Host	HTTPS	443		CFME appliance

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME appliance (SmartProxy)	ESX Hosts (if analyzing VMs through host)	SOAP over HTTPS	902		Communication from CFME appliance to hosts
CFME appliance (SmartProxy)	vCenter (if analyzing VMs through VC)	SOAP over HTTPS	902		Communication from CFME appliance to vCenters
CFME appliance(SmartProxy)	ESX Hosts (not needed for ESXi)	SSH	22		CFME appliance console access (ssh) to ESX hosts

VMware vSphere Ports Used by Infrastructure Automation

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME appliance	Hyper-V Host (VMM agent)	WinRM/RPC/NetBIOS /SMB (over TCP)	80/135/139/445		Communication from CFME appliance to Host
CFME appliance	Hyper-V Host (file transfer)	HTTPS (using BITS)	443		Communication from CFME appliance to Host
CFME appliance	VM Guest Agent (file transfer)	HTTPS (using BITS)	443		Communication from CFME appliance to VM Guest Agent
CFME appliance	VMware ESX 3.0/3.5 Host (file transfer)	SFTP	22		Communication from CFME appliance to ESX Host
CFME appliance	VMware ESXi Host (file transfer)	SSH/HTTPS (using BITS)	443		Communication from CFME appliance to ESX Host
CFME appliance	WSUS Server (data channel)	HTTP	80/443		Communication from CFME appliance to Server
CFME appliance	SQL Server database (remote)	TDS	1433		CFME appliance connectivity to the Database
CFME appliance	Load Balancer	Load balancer config provider	80/443		
CFME appliance	Hyper-V host in untrusted domain or perimeter network (File Transfer)	TCP	443		CFME appliance connectivity to the host
CFME appliance	Hyper-V Host (file transfer)	BITS	443		Communication from CFME appliance to Host
CFME appliance	VMware Web Services	WCF	443		

SCVMM Ports Used by Infrastructure Automation

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME appliance	SQL Management (*.database.windows.net)	TDS	1433		CFME appliance connectivity to the Database
CFME appliance	Upload into Storage (*.blob.core.windows.net)	HTTP/HTTPS	80/443		
CFME appliance	Service Bus Relay HTTP Mode (*.servicebus.windows.net)	SB over HTTP	80		
CFME appliance	Service Bus Pubsub over REST (*.servicebus.windows.net)	HTTPS	443		

Initiator (CFME Role if applicable)	Receiver (CFME Role if applicable)	Application	TCP Port	UDP Port	Purpose
CFME appliance	Access Control (*.accesscontrol.windows.net)	HTTPS	443		

Azure Ports Used by Infrastructure Automation

To provide your Infrastructure Automation infrastructure with an extra layer of security, use a network layer firewall to restrict port access.

Generating SSL Certificates for Your Appliance and Database

It is important to enhance the security of SSL communication of your appliances, which, depending on your setup, may include your database appliance. The appliance image ships with a default SSL certificate. It is recommended to replace this certificate with your own certificate, either signed by a trusted Certificate Authority (CA) or self-signed.

Creating a Certificate Signing Request

The first step is to determine the host name of your appliance or database appliance by running the following command:

```
$ hostname
```

The next step is to create a Certificate Signing Request (CSR) using the **openssl** command:

```
[root@ ~]# openssl req -new -newkey rsa:2048 -out appliance.csr -keyout appliance.key
```

This command generates a 2048-bit RSA private key and asks for a passphrase for the key.

```
Generating a 2048 bit RSA private key
..................+++
............................+++
writing new private key to 'appliance.key'
Enter PEM pass phrase: **********
Verifying - Enter PEM pass phrase: **********
```

The command then provides a questionnaire requesting certain details for the key. Fill out this questionnaire. Use the output of the **hostname** command to specify the **Common Name**.

For example:

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:North Carolina
Locality Name (eg, city) [Default City]:Raleigh
Organization Name (eg, company) [Default Company Ltd]:Infrastructure Automation
Organizational Unit Name (eg, section) []:Customer Content Services
Common Name (eg, your name or your server's hostname) []:$(hostname)
Email Address []:example@example.com

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Running the command produces two files:

- **appliance.key** - The private key
- **appliance.csr** - The Certificate Signing Request (CSR)

At this stage, you would send the CSR to a trusted Certificate Authority (CA) and in return they would send you a signed certificate.

Creating a Self-signed Certificate

As an alternative to obtaining a signed certificate, you can use the **appliance.key** and **appliance.csr** files to create a self-signed certificate by running the following **openssl** commands:

```
[root@ ~]# openssl rsa -in appliance.key -out server.cer.key
[root@ ~]# openssl x509 -in appliance.csr -out server.cer -req -signkey server.cer.key -days 3650
```

This produces two files:

- **server.cer.key** - The private key for your signed certificate
- **server.cer** - The self-signed certificate

Enabling Your Certificate

Despite whether you used a trusted CA or self-signed the certificate, you should now have your own certificate for your appliance.

Copy the certificate and key files to the certificate directory on the appliance:

```
[root@ ~]# cp ~/server.cer.key /var/www/miq/vmdb/certs/server.cer.key
[root@ ~]# cp ~/server.cer /var/www/miq/vmdb/certs/server.cer
```

After the certificate and key files have been copied, restart the appliance:

```
[root@ ~]# systemctl restart evmserverd
```

The appliance now uses your own certificate.

If your environment consists of multiple appliances connecting to a single database appliance, you can use your certificate and key files to set up SSL for the database connection. For more information, see Configuring the Database to use SSL.

Important:

Updates from the Red Hat Content Delivery Network might overwrite these certificate and key files. Make sure to copy your own certificate and key files to the certificate directory after performing an update to your appliance.

Note:

See also the following article for information on replacing SSL certificates in Infrastructure Automation : https://access.redhat.com/articles/449033.

Creating Custom Encryption Keys

To avoid storing passwords in plain text, Infrastructure Automation appliances use an encryption key to encode and decode passwords. Each appliance stores the key in the **/var/www/miq/vmdb/certs/v2_key**. Changing the encryption key is recommended during setting up new Infrastructure Automation appliances only.

Red Hat does not recommend changing the encryption key for an existing appliance as the ability to decrypt the password will be lost, affecting all stored passwords in Infrastructure Automation.
To generate a new encryption key:

1. Log in to the console of your master appliance as the **root** user.
2. Run the **appliance_console** command. The Infrastructure Automation appliance information screen appears.
3. Press any key to view the appliance menu.
4. Select **Generate Custom Encryption Key**.
5. A prompt asks if for confirmation to overwrite the existing key. Enter **Y**.
6. Enter **1** for **1) Create key**.
7. The appliance generates the new key. Press any key to complete this procedure.

This completes the procedure for generating the new key. If you have external Infrastructure Automation appliances, you must share this key to ensure your whole Infrastructure Automation infrastructure is using consistent encryption. Failure to use the same key results in encryption and decryption problems.

To copy an encryption key:

1. Log in to the console of an external appliance as the **root** user.
2. Run the **appliance_console** command. The Infrastructure Automation appliance information screen appears.
3. Press any key to view the appliance menu.
4. Select **Generate Custom Encryption Key**.
5. A prompt asks if for confirmation to overwrite the existing key. Enter **Y**.
6. Select **Fetch key from remote machine**.
7. Enter the hostname or IP address of the master appliance.
8. Enter the username for SSH access to the master appliance. Use the default **root** user.
9. Enter the password for SSH access to the master appliance.
10. Enter the location of the remote key. Accept the default as **/var/www/miq/vmdb/certs/v2_key**.
11. The appliance copies the new key from the remote server. Press any key to complete this procedure.

After distributing the new key, all appliances require an update to the database configuration. For all appliances, log in as the **root** user and run the following commands replacing **dbpassword** with your database password:

```
[root@{productname_short_1} ~]# fix_auth --databaseyml --hostname localhost --password dbpassword
[root@{productname_short_1} ~]# systemctl restart evmserverd
```

This completes the new encryption key generation for your Infrastructure Automation infrastructure.

Applying SCAP Standards

The Security Content Automation Protocol (SCAP) is a set of standards to assist with vulnerability management and policy compliance. Infrastructure Automation provides a set of SCAP standards to apply to your appliance. View these SCAP rules in the **/var/www/miq/vmdb/productization/appliance_console/config/scap_rules.yml** file.

To apply the SCAP standards to your appliance's server:

1. Log in to the appliance as the **root** user.
2. Enter the **appliance_console** command. The Infrastructure Automation Appliance summary screen displays.
3. Press **Enter** to manually configure settings.
4. Select **Harden Appliance Using SCAP Configuration**.
5. The appliance console displays the following:

   ```
   Harden Appliance Using SCAP Configuration

   Locking down the appliance for SCAP...
   ```

   The appliance applies the SCAP settings from the **scap_rules.yml** file.
6. When complete, press any key to return to the summary screen.

The appliance now meets the SCAP standards set in the **scap_rules.yml** file.

Database Security

Restricting Hosts Access to the Database

Strengthening the host-based authentication (HBA) settings on a database appliance helps with preventing unauthorized access from external hosts. The HBA settings restrict access to an IP address range so that only hosts within that range have access.

Restricting access to the database requires modifications to the **/var/lib/pgsql/data/pg_hba.conf** file. This file contains a text-based table with some initial settings:

```
# TYPE    DATABASE USER  ADDRESS       METHOD
local     all      all                  peer map=usermap
host      all      all   all            md5
#hostssl  all      all   all            md5
```

This format for this table uses the following header columns:

- TYPE This defines the access type, either local access from the database host (**local**), remote access from an external host regardless of encryption (**host**), external access with encryption (**hostssl**), or external access without encryption (**nohostssl**).
- DATABASE The name of the database the host can access. Use **all** for all databases.
- USER The name of the user the host can use to access the database. Use **all** for all users.
- ADDRESS The IP address of the host or address range of hosts with access to the database. This can either be:
  - A single address:

    ```
    host    all      all   192.168.1.10          md5
    ```
  - An address range using a CIDR mask:

    ```
    host    all      all   192.168.1.0/24          md5
    ```
  - An address range using a separate subnet mask value

    ```
    host    all      all   192.168.1.0  255.255.255.0          md5
    ```

    Note:

    ADDRESS is not required for **local** connections.

- METHOD The authentication method, which includes:
  - **trust** - Allow the connection unconditionally. This method allows anyone that can connect to the PostgreSQL database server to login as any PostgreSQL user they wish, without the need for a password or any other authentication.
  - **reject** - Reject the connection unconditionally. This is useful for “filtering out” certain hosts from a group, for example a **reject** line could block a specific host from connecting, while a later line allows the remaining hosts in a specific network to connect.
  - **md5** - Require the client to supply an MD5-encrypted password for authentication.
  - **password** - Require the client to supply an unencrypted password for authentication. Since the password is sent in clear text over the network, this should not be used on untrusted networks.
  - **ident** - Obtain the operating system user name of the client by contacting the ident server on the client and check if it matches the requested database user name. Ident authentication can only be used on TCP/IP connections. When specified for local connections, peer authentication will be used instead.
  - **peer** - Obtain the client’s operating system user name from the operating system and check if it matches the requested database user name. This is only available for local connections.

Using a combination of these options, you create a series of rules that govern which hosts can access your database and which hosts are denied. For example, you might change the default HBA rules to only allow remote access to the Infrastructure Automation database (**vmdb_production**) from hosts in a certain subnet. The modified HBA table would looks like this:

```
# TYPE  DATABASE            USER  ADDRESS          METHOD
local   all                 all                    peer map=usermap
host    vmdb_production     all   192.168.1.0/24   md5
#hostssl all                all   all              md5
```

These restrictions help when structuring your Infrastructure Automation appliances in relationships. For example, use these database restrictions to grant access only between a master database appliance in one region and appliances connecting from a separate region.

Configuring the Database to use SSL

Infrastructure Automation initially connects to the database through an unencrypted communication. If you are using multiple appliances that are connecting to a single database appliance, you can set up the database connection to use SSL. An SSL connection encrypts the communication between the Infrastructure Automation and the database.

The procedures in this section use the SSL certificate and the following key files. These files can be found on your main Infrastructure Automation database appliance.

Note:

The appliance image ships with a default SSL certificate and it is recommended to change this certificate. You can use a certificate that is signed by a trusted CA, or generate a self-signed certificate.

For more information, see Generating SSL Certificates for Your Appliance and Database.

- **/var/www/miq/vmdb/certs/server.cer** - Signed or self-signed certificate for the database appliance.
- **/var/www/miq/vmdb/certs/server.cer.key** - Private key for server certificate.

It is also recommended to stop all Infrastructure Automation services before configuring the database to use SSL.

Configure SSL on the database appliance

To configure SSL on the database appliance:

1. Log in as **root** to the appliance where the database resides.
2. Stop the **evmserverd** and **postgresql** services:

   ```
   $ systemctl stop evmserverd
   $ systemctl stop postgresql.service
   ```

3. Install the server key file in the correct location and set the ownership and permissions for it:

   ```
   $ install -m 600 -o postgres -g postgres \
   /var/www/miq/vmdb/certs/server.cer.key /var/lib/pgsql/data/postgres.key
   ```

4. Install the server certificate file in the correct location and set the ownership and permissions for it:

   ```
   $ install -m 644 -o postgres -g postgres \
   /var/www/miq/vmdb/certs/server.cer /var/lib/pgsql/data/postgres.crt
   ```

5. Open the **/var/lib/pgsql/data/postgresql.conf** file and uncomment and edit the **ssl** option:

```
ssl=on
```

In the same file, locate the options **ssl_cert_file** and **ssl_key_file** that specify the location of SSL certificates and edit them so that they are uncommented and point to the correct certificate files:

```
ssl_cert_file = 'postgres.crt'   # (change requires restart)
ssl_key_file  = 'postgres.key'   # (change requires restart)
```

6. Open the **/var/lib/pgsql/data/pg_hba.conf** file and locate the two lines that contain:

```
host      all      all   all           md5
#hostssl  all      all   all           md5
```

Modify the two lines to comment the **host** entry and uncomment the **hostssl** entry:

```
#host     all      all   all           md5
hostssl   all      all   all           md5
```

This changes the incoming communication protocol to use SSL and refuse any unencrypted PostgreSQL connections.

7. Start the **postgresql** and **evmserverd** services so that the changes take effect:

```
$ systemctl start postgresql.service
$ systemctl start evmserverd
```

Configure SSL on all connected appliances

The database appliance now accepts only connections from connecting appliances that use SSL. The following procedure sets up connecting appliances to communicate to the database by using SSL. Use this procedure for each connecting appliance:

1. Log in as **root** to the connecting appliance.

2. Copy the root certificate file from the database to the operating system's list of valid certificates:

```
$ DATABASE=[database_appliance_fqdn]
$ scp root@${DATABASE}:/var/www/miq/vmdb/certs/root.crt /etc/pki/ca-trust/source/anchors/${DATABASE}-
postgres
$ update-ca-trust extract
$ export PGSSLROOTCERT=/etc/pki/tls/certs/ca-bundle.crt
```

Where **[database_appliance_fqdn]** is the fully qualified domain name of the database appliance.

3. Test the certificate got added to the certificate database:

```
$ openssl x509 -in /etc/pki/tls/certs/ca-bundle.crt -text -noout
```

4. Test the connection between the connecting appliance and the database appliance by using the **psql**command:

```
$ PGSSLMODE=verify-full psql -h [database_appliance_fqdn] -d vmdb_production \
  -c "SELECT * from pg_stat_ssl WHERE pid=pg_backend_pid()"
Password: ********
psql (9.2.8)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

vmdb_production=#
```

The **psql** displays information about the SSL connection, which indicates that the configuration succeeded. Enter **\q** to leave **psql**. The **PGSSLMODE** variable forces a certain level of SSL. To learn more, please see postgress sslmode documentation.

5. Ensure **/etc/default/manageiq.properties** has the proper ssl key file location defined:

```
$ grep PGSSLROOTCERT /etc/default/manageiq.properties || \
echo PGSSLROOTCERT=/etc/pki/tls/certs/ca-bundle.crt >> /etc/default/manageiq.properties
```

The standard **manageiq.properties** should already have the **PGSSLROOTCERT** defined. The above command adds an entry to **manageiq.properties** if it is not already present.

Complete this procedure for each external appliance. This enhances the security of all database transactions in your Infrastructure Automation infrastructure.

Hardening TLS Protocol Version

After configuring the database to use SSL, protocol TLS version 1.2 is used as default. The older versions of this protocol (TLS 1.0 and 1.1) are still available for clients to choose. You can disable older versions by inserting the following lines into **/var/lib/pgsql/data/postgresql.conf**:

```
ssl_ciphers = 'TLSv1.2:!aNULL'
ssl_prefer_server_ciphers=true
```

- Methods Available for Automation
  - Methods Available for Use with Infrastructure Automation
  - $evm.root
    - Method Hierarchy
    - Base Methods
    - Availability Zones (availability_zone)
    - Classifications (classification)
    - Cloud Networks (cloud_network)
    - Cloud Resource Quotas (cloud_resource_quota)
    - Cloud Subnets (cloud_subnet)
    - Cloud Tenants (cloud_tenant)
    - Customization Template (customization_template)
    - Clusters (ems_cluster)
    - Management System Events (ems_event)
    - Management System Folders (ems_folder)
    - Providers (ext_management_system)
      - Cloud Providers (ems_cloud)
    - Firewall Rules (firewall_rule)
    - Flavors (flavor)
    - Floating IPs (floating_ip)
    - Guest Applications (guest_application)
    - Guest Device (guest_device)
    - Hardware (hardware)
    - Hosts (host)
      - Hosts: Vmware ESX (host_vmware_esx)
    - LAN (lan)
    - Groups (miq_group)
    - Request (miq_request)
      - Automation Request (automation_request)
      - Host Provision Request (miq_host_provision_request)
      - VM Provision Request (miq_provision_request)
    - Request Task (miq_request_task)
      - Automation Task (automation_task)
      - Host Provision Task (miq_host_provision)
      - VM Provision Task (miq_provision)
        - Adding Disks to OpenStack Instance Provisioning
        - Booting OpenStack Instances from Volume
        - VM Provisioning for Clouds Task (miq_provision_cloud)
      - Service Template Provision Task (service_template_provision_task)
      - Service Reconfiguration Task (service_reconfigure_task)
      - VM Migrate Task (vm_migrate_task)
    - Proxies (miq_proxy)
    - Servers (miq_server)
    - Network (network)
    - PXE Image (pxe_image)
    - PXE Server (pxe_server)
    - Security Groups (security_group)
    - Service (service)
    - Service Resource (service_resource)
    - Service Template (service_template)
    - Snapshot (snapshot)
    - Datastores (storage)
    - Switch (switch)
    - User (user)
    - Virtual Machines and Templates (vm_or_template)
      - VMs (vm)
        - VMs for Clouds (vm_cloud)
    - Windows Image (windows_image)
    - Creating Categories and Tags
      - Category and Tags Methods Example
    - Quota
- Terminology

Methods Available for Automation

Methods Available for Use with Infrastructure Automation

Methods can be used from within Infrastructure Automation to create custom actions and workflows for the objects managed for your Infrastructure Automation Infrastructure. This document describes the methods available for use in Infrastructure Automation. This document is organized by the object hierarchy in the Automate Model.

Note:

Users of Infrastructure Automation can construct custom automation methods in Ruby to extend the product. Infrastructure Automation ships with a core set of Ruby gems used by the Infrastructure Automation Rails Application. The Ruby gems in this set are subject to change, and have changed since the previous version. If you are calling gems using Automate that are no longer in the Infrastructure Automation appliance, you can install them by using the **`gem install`** command.

While gems can be imported into automation methods using **`require`**, it is recommended that the authors of the automation methods clearly document the use of gems either in the core set or a custom set. It is the responsibility of the author of such custom automation to own the life cycle of any gem being referenced in those methods.

$evm.root

When an Automate method is launched, it has one global variable: **`$evm`**. The **`$evm`** variable allows the method to communicate back to Infrastructure Automation. The **`$evm.root`** is the root object in the workspace, it provides access to the data currently loaded in the Infrastructure Automation model. It uses the object's data to solve more complex problems by integrating with Infrastructure Automation methods.

The following is an excerpt from the **InspectMe** method that can be found in the **ManageIQ\System\Request** namespace. The dumpRoot method accesses the **`$evm.root`** object, and sends all of its attributes to the Infrastructure Automation Automate log for review. In the dumpServer Method, the inspect method is run based on the value of the miq_server obtained from the **`$evm.root`** object.

```
#########################
  #
  # Method: dumpRoot
  # Description: Dump Root information
  #
  #########################
  def dumpRoot
    $evm.log("info", "#{@log_prefix} - Root:<$evm.root> Begin Attributes")
    $evm.root.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - Root:<$evm.root> Attributes -
#{k}: #{v}")}
    $evm.log("info", "#{@log_prefix} - Root:<$evm.root> End Attributes")
    $evm.log("info", "")
  end

  #########################
  #
  # Method: dumpServer
  # Inputs: $evm.root['miq_server']
  # Description: Dump MIQ Server information
  #
  #########################
  def dumpServer
    $evm.log("info","#{@log_prefix} - Server:<#{$evm.root['miq_server'].name}> Begin Attributes")
    $evm.root['miq_server'].attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - Server:<#
{$evm.root['miq_server'].name}> Attributes - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Server:<#{$evm.root['miq_server'].name}> End Attributes")
    $evm.log("info", "")
  end
```

The result of dumpRoot is below. The value of miq_server is what gets passed into the dumpServer method.

```
[----] I, [2012-10-23T13:53:54.517279 #5320:f329024]  INFO -- : <User-Defined Method> [InspectMe] - EVM
Automate Method Started
[----] I, [2012-10-23T13:53:54.523637 #5320:f329024]  INFO -- : <User-Defined Method> [InspectMe] - Root:
<$evm.root> Begin Attributes
[----] I, [2012-10-23T13:53:54.527552 #5320:ef8c538]  INFO -- : <User-Defined Method> [InspectMe] - Root:
<$evm.root> Attributes - miq_server: #<MiqAeMethodService::MiqAeServiceMiqServer:0x0000001e76d900>
[----] I, [2012-10-23T13:53:54.528801 #5320:ef8c538]  INFO -- : <User-Defined Method> [InspectMe] - Root:
<$evm.root> Attributes - miq_server_id: 1
[----] I, [2012-10-23T13:53:54.529961 #5320:ef8c538]  INFO -- : <User-Defined Method> [InspectMe] - Root:
<$evm.root> Attributes - object_name: Request
[----] I, [2012-10-23T13:53:54.531067 #5320:ef8c538]  INFO -- : <User-Defined Method> [InspectMe] - Root:
```

```
<$evm.root> Attributes - request: inspectme
[----] I, [2012-10-23T13:53:54.534054 #5320:ef8c538]  INFO -- : <User-Defined Method> [InspectMe] - Root:
<$evm.root> Attributes - vm: DEV-JaneM
[----] I, [2012-10-23T13:53:54.535156 #5320:ef8c538]  INFO -- : <User-Defined Method> [InspectMe] - Root:
<$evm.root> Attributes - vm_id: 85
[----] I, [2012-10-23T13:53:54.536238 #5320:ef8c538]  INFO -- : <User-Defined Method> [InspectMe] - Root:
<$evm.root> Attributes - vmdb_object_type: vm
[----] I, [2012-10-23T13:53:54.537159 #5320:f329024]  INFO -- : <User-Defined Method> [InspectMe] - Root:
<$evm.root> End Attributes
[----] I, [2012-10-23T13:53:54.537772 #5320:f329024]  INFO -- : <User-Defined Method>
```

Method Hierarchy

The Automate Model inline methods have a hierarchy. The sublevels in the hierarchy have access to the methods for itself and the levels above it. For example, Red Hat Hosts have access to the Red Hat Host methods, Host Methods, and Base Methods.

The following nested list displays the hierarchy.

- Top Level: Base
    - Authentication (authentication)
        - Private Keys (auth_private_key)
            - Key Pair for Clouds (auth_key_pair_cloud)
                - Amazon (auth_key_pair_amazon)
                - OpenStack (auth_key_pair_openstack)
    - Availability Zones (availability_zone)
        - Amazon (availability_zone_amazon)
        - OpenStack (availability_zone_openstack)
    - Classification (classification)
    - Cloud Networks (cloud_network)
    - Cloud Resource Quotas (cloud_resource_quota)
        - OpenStack (openstack_resource_quota)
    - Cloud Subnets (cloud_subnet)
    - Cloud Tenants (cloud_tenant)
    - Customization Templates (customization_template)
        - Cloud Init (customization_template_cloud_init)
        - Kickstart (customization_template_kickstart)
        - Sysprep (customization_template_sysprep)
    - Cluster (ems_cluster)
    - Event (ems_event)
    - Folder (ems_folder)
    - Providers (ext_management_system)
        - Cloud (ems_cloud)
            - Amazon (ems_amazon)
            - Openstack (ems_openstack)
        - Infrastructure (ems_infra)
            - Microsoft System Center VMM (ems_microsoft)
            - Red Hat Enterprise Virtualization (ems_redhat)
            - VMware vCenter (ems_vmware)
    - Filesystems (filesystem)

- Firewall Rules (filewall_rule)
- Flavors
    - Amazon (flavor_amazon)
    - OpenStack (flavor_openstack)
- Floating IPs (floating_ip)
    - Amazon (floating_ip_amazon)
    - OpenStack (floating_ip_openstack)
- Guest Applications (guest_application)
- Guest Devices (guest_device)
- Hardware (hardware)
- Hosts (host)
    - Red Hat Enterprise Virtualization (host_redhat)
    - VMware (host_vmware)
        - VMware ESX (host_vmware_esx)
- ISO Images (iso_image)
- Jobs (job)
- LANs (lan)
- Groups (miq_group)
- Policies (miq_policy)
- Proxies (miq_proxy)
- Requests (miq_request)
    - Automation (automation_request)
    - Host Provisioning (miq_host_provision_request)
    - VM Provisioning (miq_provision_request)
        - VM Templates (miq_provision_request_template)
    - Service Reconfiguration (service_reconfigure_request)
    - Service Template Provisioning (service_template_provision_request)
    - VM Migration (vm_migrate_request)
    - VM Reconfiguration (vm_reconfigure_request)
- Request Task (miq_request_task)
    - Automation (automation_task)
    - Host Provisioning (miq_host_provision)
    - VM Provisioning (miq_provision)
        - Cloud (miq_provision_cloud)
            - Amazon (miq_provision_amazon)
            - OpenStack (miq_provision_openstack)
        - Red Hat Enterprise Virtualization (miq_provision_redhat)
            - Via ISO (miq_provision_redhat_via_iso)
            - Via PXE (miq_provision_redhat_via_pxe)
        - VMware (miq_provision_vmware)

                - Via NetApp RCU (miq_provision_vmware_via_net_app_rcu)
                - Via PXE (miq_provision_vmware_via_pxe)
            - Service Reconfiguration (service_reconfigure_task)
            - Service Template Provisioning (service_template_provision_task)
            - VM Migratation (vm_migrate_task)
            - VM Reconfiguration (vm_reconfigure_task)
    - Servers (miq_server)
    - Networks (network)
    - Operating Systems (operating_system)
    - PXE Images (pxe_image)
        - iPXE (pxe_image_ipxe)
        - PXELINUX (pxe_image_pxelinux)
    - PXE Servers (pxe_server)
    - Resource Pools (resource_pool)
    - Security Groups (security_group)
        - Amazon (security_group_amazon)
        - OpenStack (security_group_openstack)
    - Services (service)
    - Service Resources (service_resource)
    - Service Templates (service_template)
    - Snapshots (snapshot)
    - Storages (storage)
    - Switches (switch)
    - Users (user)
    - VMs or Templates (vm_or_template)
        - Templates (miq_template)
            - Cloud (template_cloud)
                - Amazon (template_amazon)
                - OpenStack (template_openstack)
            - Infrastructure (template_infra)
                - Microsoft (template_microsoft)
                - Red Hat Enterprise Virtualization (template_redhat)
                - VMware (template_vmware)
        - VMs (vm)
            - Clouds (vm_cloud)
                - Amazon (vm_amazon)
                - OpenStack (vm_openstack)
            - Infrastructure (vm_infra)
                - Microsoft (vm_microsoft)

        - Red Hat Enterprise Virtualization (vm_redhat)
        - Vmware (vm_vmware)
    - Windows Images (windows_images)

Base Methods

These methods may be used with all objects available in the Automate Model.

Method	Usage
inspect	Returns a string containing a list of attributes of the object. See the InspectMe method in Samples class
inspect_all	Returns all information for an object
virtual_column_names	Returns the objects virtual columns names
virtual_columns_inspect	Returns the objects virtual columns and values
reload	Returns to original object to prevent the internal object from being returned
model_suffix	Returns objects suffix. For an object of type MiqAeServiceVmVmware, returns “Vmware”
tagged_with?(category, name)	Is the object tagged with the category and name specified?
tags(category = nil)– this means that category is an optional parameter, with a default of nil	Returns the tags.
tag_assign(tag)	Assigns tag to the object, except for the miq_provision object, which uses add_tag(category, tag_name)
tag_unassign(tag)	Unassigns tag to the object, except for the miq_provision object, which uses clear_tag(category, tag_name)

The **InspectMe Sample Method** uses many of the Methods shown in this document. The method returns attributes of the Infrastructure Automation Server and then returns attributes for the host, cluster, and virtual machine from the provider of invocation. In many environments it is linked to a button.

```
###################################
# EVM Automate Method: InspectMe
#
# Notes: Dump the objects in storage to the automation.log
#
###################################
begin
  @method = 'InspectMe'
  @log_prefix = "[#{@method}]"
  $evm.log("info", "#{@log_prefix} - EVM Automate Method Started")

  # Turn on verbose logging
  @debug = true



  # List the types of object we will try to detect
  obj_types = %w{ vm host storage ems_cluster ext_management_system }
  obj_type = $evm.root.attributes.detect { |k,v| obj_types.include?(k)}

  # uncomment below to dump root object attributes
  dumpRoot

  # uncomment below to dump miq_server object attributes
  dumpServer


  # If obj_type is NOT nil
  unless obj_type.nil?
    rootobj = obj_type.first
    obj = obj_type.second
    $evm.log("info", "#{@log_prefix} - Detected Object:<#{rootobj}>")
    $evm.log("info","")

    case rootobj
    when 'host' then dumpHost(obj)
    when 'vm' then dumpVM(obj)
    when 'ems_cluster' then dumpCluster(obj)
    when 'ext_management_system' then dumpEMS(obj)
    when 'storage' then dumpStorage(obj)
    end
  end

  #
  # Exit method
```

```
  #
  $evm.log("info", "#{@log_prefix} - EVM Automate Method Ended")
  exit MIQ_OK

  #
  # Set Ruby rescue behavior
  #
rescue => err
  $evm.log("error", "#{@log_prefix} - [#{err}]\n#{err.backtrace.join("\n")}")
  exit MIQ_ABORT
end
```

Availability Zones (availability_zone)

Method	Use
ext_management_system	Returns object's Management System
vms	Returns object's VMs
vms_and_templates	Returns object's VMs and templates
cloud_subnets	Returns object's cloud subnets

Classifications (classification)

Method	Use
parent	Returns object's parent object
namespace	Returns object's namespace
category	Returns object's category
name	Returns object's name
to_tag	Returns object's tag mapping

Cloud Networks (cloud_network)

Method	Use
ext_management_system	Returns object's Management System
cloud_tenant	Returns object's cloud tenant
cloud_subnets	Returns object's cloud subnets
security_groups	Returns object's security groups
vms	Returns object's VMs

Cloud Resource Quotas (cloud_resource_quota)

Method	Use
ext_management_system	Returns object's Management System
cloud_tenant	Returns object's cloud tenant

Cloud Subnets (cloud_subnet)

Method	Use
cloud_network	Returns object's cloud network
availability_zone	Returns object's availability zone
vms	Returns object's VMs

Cloud Tenants (cloud_tenant)

Method	Use
ext_management_system	Returns object's Management System
security_groups	Returns object's security groups
cloud_networks	Returns object's cloud network
vms	Returns object's VMs
vms_and_templates	Returns object's VMs and templates
miq_templates	Returns object's templates
floating_ips	Returns object's floating IP addresses
cloud_resource_quotas	Returns object's quotas

Customization Template (customization_template)

Method	Use
Pxe_images	Returns customization templates pxe images

Clusters (ems_cluster)

Method	Use
all_resource_pools	Return all of the objects Resource Pools
all_vms	Return all of the objects Virtual Machines
default_resource_pool	Return the objects default Resource Pool
ems_events	Returns an array of EmsEvent records associated with the object
ext_management_system	Return objects Management System
hosts	Return objects Hosts
parent_folder	Return objects Parent Folder
register_host(host)	Register Host to this Cluster
resource_pools	Return objects Resource Pools
storages	Return objects datastores
vms	Return objects Virtual Machines

```
  ##########################
  #
  # Method: dumpCluster
  # Inputs: $evm.root['ems_cluster']
  # Description: Dump Cluster information
  #
  ##########################
  def dumpCluster(cluster)
    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> Begin Attributes")
    cluster.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - Cluster:<#{cluster.name}>
Attributes - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> End Attributes")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> Begin Associations")
    cluster.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix} - Cluster:<#{cluster.name}>
Associations - #{assc}")}
    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> End Associations")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> Begin Virtual Columns")
    cluster.virtual_column_names.sort.each { |vcn| $evm.log("info", "#{@log_prefix} - Cluster:<#
{cluster.name}> Virtual Columns - #{vcn}: #{cluster.send(vcn)}")}
    $evm.log("info","#{@log_prefix} - Cluster:<#{cluster.name}> End Virtual Columns")
    $evm.log("info","")
  end
```

Management System Events (ems_event)

Method	Use
ext_management_system	Returns object's provider
ems	Shortcut to ext_management_system
src_vm	Source VM for the event
vm	VM for the event
src_host	Source Host for the event
host	Host for the event
dest_vm	Destination VM for the event
service	Service for the event
dest_host	Destination Host for the event
refresh(*targets)	Refresh the target types specified (ems, vm, host, src_vm, src_host, dest_vm, or dest_host)

Management System Folders (ems_folder)

Method	Use
hosts	Returns hosts that are in the folder
vms	Returns VMs that are in folder
register_host(host)	Registers specified host to the folder
folder_path(*options)	Returns folders path

Providers (ext_management_system)

Method	Use
authentication_password_encrypted	Returns credentials password encrypted
authentication_password	Returns credentials password unencrypted

Method	Use
authentication_userid	Returns credentials user id
ems_clusters	Returns objects clusters
ems_events	Returns an array of EmsEvent records associated with the object
ems_folders	Returns objects folders
hosts	Returns objects hosts
refresh	Refreshes relationships and power states for objects related to the object
resource_pools	Returns objects resource pools
storages	Returns objects storages
vms	Returns objects vms
to_s	Converts object to string

```
  ############################
  #
  # Method: dumpEMS
  # Inputs: $evm.root['ext_management_system']
  # Description: Dump EMS information
  #
  ############################
  def dumpEMS(ems)
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> Begin Attributes")
    ems.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - EMS:<#{ems.name}> Attributes - #{k}:
#{v.inspect}")}
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> End Attributes")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> Begin Associations")
    ems.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix} - EMS:<#{ems.name}> Associations - #
{assc}")}
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> End Associations")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> Begin EMS Folders")
    ems.ems_folders.each { |ef| ef.attributes.sort.each { |k,v| $evm.log("info", "#{@log_prefix} - EMS:<#
{ems.name}> EMS Folder:<#{ef.name}> #{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> End EMS Folders")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> Begin Virtual Columns")
    ems.virtual_column_names.sort.each { |vcn| $evm.log("info", "#{@log_prefix} - EMS:<#{ems.name}> Virtual
Columns - #{vcn}: #{ems.send(vcn)}")}
    $evm.log("info","#{@log_prefix} - EMS:<#{ems.name}> End Virtual Columns")
    $evm.log("info","")
  end
```

Cloud Providers (ems_cloud)

Method	Use
availability_zones	Return the provider's availability zones
cloud_networks	Return the provider's available networks
cloud_tenants	Return the provider's available tenants
flavors	Return the provider's hardware flavors
floating_ips	Return the provider's floating IP addresses
key_pairs	Return the provider's key pairs
security_groups	Return the provider's security groups
cloud_resource_quotas	Return the provider's resource quotas

Firewall Rules (firewall_rule)

Method	Use
resource	Return object's resource
source_security_group	Return object's source security group

Flavors (flavor)

Method	Use
ext_management_system	Returns object's Management System
vms	Returns object's VMs

Floating IPs (floating_ip)

Method	Use
ext_management_system	Returns object's Management System
vm	Returns object's VMs
cloud_tenant	Returns object's cloud tenant

Guest Applications (guest_application)

Method	Use
vm	Returns objects VM
host	Returns objects Host

Guest Device (guest_device)

Method	Use
hardware	Returns objects hardware
switch	Returns objects switch
lan	Returns objects LAN
network	Returns objects network

Hardware (hardware)

Method	Use
ipaddresses	Returns objects IP addresses
guest_devices	Returns objects guest devices
storage_adapters	Returns objects storage adapters
nics	Returns objects nics
ports	Returns objects ports
vm	Returns objects Virtual Machine
host	Returns objects Host
mac_addresses	Returns objects MAC addresses

Hosts (host)

Method	Use
authentication_password	Returns credential password
authentication_userid	Returns credential user
datacenter	Returns datacenter
directories	Returns list of directories for the object
domain	Returns the domain portion of the hostname
ems_cluster	Returns cluster
ems_events	Returns an array of EmsEvent records associated with the object
ems_folder	Returns hosts folder on Management System
event_log_threshold?(options)	Searches event log records to determine if an event has occurred x number of times within a defined time frame. Returns true if the number of matching records found are greater or equal to the specified freq_threshold, otherwise it returns false Options values: :message_filter_type - Must be one of "STARTS WITH", "ENDS WITH", "INCLUDES", "REGULAR EXPRESSION" :message_filter_value - <string value to search for> :time_threshold - Options time interval to search. Example: 2.days (Search the past 2 days of event logs) Default 10.days :freq_threshold - Number of occurrences to check for. Default = 2 :source, :event_id, :level, :name - Options filter values
ext_management_system	Returns Management System
files	Returns list of files for the object
guest_applications	Returns Guest Applications
hardware	Returns hardware
lans	Returns LANs

Method	Use
operating_system	Returns Operating System
storages	Returns datastores
switches	Returns network switches
vms	Returns VMs.
credentials(type = :remote)	Returns credentials for a Host for the specified type as an array for username/pwd. (Default type is :remote if no type is specified.) Supports 4 different types of credentials: :default = Default :remote = Remote Login (think SSH for ESX) :ws = Web Services :ipmi = IPMI Example 1: cred = host.credentials cred ⇒ ["user", "pwd"] Example 2: user_str, pwd_str = host.credentials(:ipmi) user_str ⇒ "user" pwd_str ⇒ "pwd"
ems_custom_keys	Returns Management Systems custom keys
ems_custom_get(key)	Gets Value for specified Management Systems custom key
ems_custom_set(attribute, value)	Sets value for specified custom key of the Management System
custom_keys	Lists Infrastructure Automation Server custom keys
custom_get(key)	Gets value for specified Infrastructure Automation Server custom key
custom_set(key, value)	Sets value for specified Infrastructure Automation Server custom key
ssh_exec(script)	Runs the specified script on the host
get_realtime_metric(metric, range, function)	Returns specified realtime metric
current_memory_usage	Returns current memory usage
current_cpu_usage	Returns current cpu usage
current_memory_headroom	Returns current memory headroom
to_s	Converts object to string
scan	Performs SmartState Analysis on the object

The following table lists the metric types available for the **`get_realtime_metric(metric, range, function)`** method for hosts.

Metric	Description
v_derived_storage_used	Capacity - Used space in bytes
v_pct_cpu_ready_delta_summation	CPU - Percentage ready
v_pct_cpu_wait_delta_summation	CPU - Percentage wait
v_pct_cpu_used_delta_summation	CPU - Percentage used
v_derived_host_count	State - Number of hosts (Hourly Count / Daily Average)
v_derived_cpu_reserved_pct	CPU - Percentage available
v_derived_memory_reserved_pct	Memory - Percentage available

The following Ruby snippet demonstrates using the **`get_realtime_metric(metric, range, function)`** method using the **`v_pct_cpu_ready_delta_summation`** metric.

```
host = $evm.root['host']
cpu_rdy = host.get_realtime_metric(:v_pct_cpu_ready_delta_summation, [15.minutes.ago.utc,5.minutes.ago.utc], :avg)
```

```
#########################
  #
  # Method: dumpHost
  # Inputs: $evm.root['host']
  # Description: Dump Host information
  #
  #########################
  def dumpHost(host)
    host = $evm.object['host'] || $evm.root['host']
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Attributes")
    host.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - Host:<#{host.name}> Attributes - #
{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Attributes")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Associations")
    host.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix} - Host:<#{host.name}> Associations
- #{assc}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Associations")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Hardware")
    host.hardware.attributes.each { |k,v| $evm.log("info", "#{@log_prefix} - Host:<#{host.name}> Hardware - #
{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Hardware")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Lans")
    host.lans.each { |lan| lan.attributes.sort.each { |k,v| $evm.log("info", "#{@log_prefix} - Host:<#
{host.name}> Lan:<#{lan.name}> - #{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Lans")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Switches")
    host.switches.each { |switch| switch.attributes.sort.each { |k,v| $evm.log("info", "#{@log_prefix} -
Host:<#{host.name}> Swtiche:<#{switch.name}> - #{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Switches")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Operating System")
    host.operating_system.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - Host:<#
{host.name}> Operating System - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Operating System")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Guest Applications")
    host.guest_applications.each { |guest_app| guest_app.attributes.sort.each { |k, v| $evm.log("info", "#
{@log_prefix} - Host:<#{host.name}> Guest Application:<#{guest_app.name}> - #{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Guest Applications")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> Begin Virtual Columns")
    host.virtual_column_names.sort.each { |vcn| $evm.log("info", "#{@log_prefix} - Host:<#{host.name}>
Virtual Columns - #{vcn}: #{host.send(vcn).inspect}")}
    $evm.log("info","#{@log_prefix} - Host:<#{host.name}> End Virtual Columns")
    $evm.log("info", "")
  end
```

Hosts: Vmware ESX (host_vmware_esx)

Method	Use
disable_vmotion(device = nil)	Disable vMotion
enable_vmotion(device = nil)	Enable vMotion
enter_maintenance_mode(timeout = 0, evacuate = false)	Put Host in Maintenance Mode
exit_maintenance_mode(timeout = 0)	Leave Maintenance Mode
in_maintenance_mode?	Check to see if the host is in Maintenance Mode
power_down_to_standby(timeout = 0, evacuate = false)	Put Host in standby
power_up_from_standby(timeout = 0)	Take Host out of standby
reboot(force = false)	Reboot Host
shutdown(force = false)	Shutdown Host
vmotion_enabled?(device = nil)	Check to see if vMotion is enabled

LAN (lan)

Method	Use
switch	Returns objects switch

Method	Use
guest_devices	Returns objects guest devices
vms	Returns objects Virtual Machines
templates	Returns objects templates
hosts	Returns objects Hosts

Groups (miq_group)

Method	Use
users	Returns users in the current miq_group
vms	Returns Virtual Machines that this group owns
custom_keys	Returns all custom keys for the group
custom_get(key)	Returns the value of the specified custom key for the group
custom_set(key, value)	Sets the value for the specified key

```
  ##########################
  #
  # Method: dumpGroup
  # Inputs: $evm.root['user'].miq_group
  # Description: Dump User's Group information
  #
  ##########################
  def dumpGroup
    user = $evm.root['user']
    unless user.nil?
      miq_group = user.miq_group
      unless miq_group.nil?
        $evm.log("info","#{@method} - Group:<#{miq_group.description}> Begin Attributes
[miq_group.attributes]")
        miq_group.attributes.sort.each { |k, v| $evm.log("info", "#{@method} - Group:<#
{miq_group.description}> Attributes - #{k}: #{v.inspect}")} unless $evm.root['user'].miq_group.nil?
        $evm.log("info","#{@method} - Group:<#{miq_group.description}> End Attributes
[miq_group.attributes]")
        $evm.log("info", "")

        $evm.log("info","#{@method} - Group:<#{miq_group.description}> Begin Associations
[miq_group.associations]")
        miq_group.associations.sort.each { |assc| $evm.log("info", "#{@method} - Group:<#
{miq_group.description}> Associations - #{assc}")}
        $evm.log("info","#{@method} - Group:<#{miq_group.description}> End Associations
[miq_group.associations]")
        $evm.log("info","")

        $evm.log("info","#{@method} - Group:<#{miq_group.description}> Begin Virtual Columns
[miq_group.virtual_column_names]")
        miq_group.virtual_column_names.sort.each { |vcn| $evm.log("info", "#{@method} - Group:<#
{miq_group.description}> Virtual Columns - #{vcn}: #{miq_group.send(vcn).inspect}")}
        $evm.log("info","#{@method} - Group:<#{miq_group.description}> End Virtual Columns
[miq_group.virtual_column_names]")
        $evm.log("info","")
      end
    end
  end
```

Request (miq_request)

Request objects are submitted to Infrastructure Automation Server for processing. After the request phase, the request becomes a task object. The table below shows the relationship between a request object and a task object.

Request Object	Task Object
automation_request	automation_task
miq_host_provision_request	miq_host_provision
miq_provision_request	miq_provision
vm_reconfigure_request	vm_reconfigure_task
service_template_provision_request	service_template_provision_task
vm_migrate_request	vm_migrate_task

If you set something on the request object, it will be inherited by the task instance that does the work. This may be useful if you are provisioning multiple virtual machines at a time and need to modify the same setting for all. Otherwise, the item can be modified on the individual task.

Method	Use
add_tag(category, tag_name)	Add a tag to a provision instance by specifying the category and tag name Example: miq_provision.add_tag(:location, "CHI")

Method	Use
approve(approver,reason for approval)	Approves the miq_request. Example: $evm.root["miq_request"].approve("admin", "Auto-Approved")
approvers	Returns request approvers.
authorized?	Returns true if authorized, false if not.
clear_tag(category=nil, tag_name=nil)	Without any parameters, the clear_tag method will clear all tags from the provision request. Providing a category will clear all tags selected in that category. Clear a specific category/tag by providing it. Example: miq_provision.clear_tag(:location, "CHI")
deny	Denies the miq_request Example: # Deny the request $evm.log('info',"Request denied because of Quota") $evm.root["miq_request"].deny("admin", "Quota Exceeded")
get_classification(category)	Works the same as get_tag(category) but the returned data is a hash with :name and :description Example: request.get_classification(:department) Returns: [{:name⇒"accounting", :description⇒"Accounting"}, {:name⇒"engineering", :description⇒"Engineering"}]
get_classifications	Works the same as get_tag but the returned tag data is a hash with :name and :description Example: request.get_classifications Returns: {:cc⇒{:name⇒"001", :description⇒"Cost Center 001"}, :department⇒[{:name⇒"accounting", :description⇒"Accounting"}, {:name⇒"engineering", :description⇒"Engineering"}]}
get_option(key)	Returns a value from the options hash based on the name of the key name passed in. Internally many of the values are stored as an array of items. (For example, a target host would be stored as the index to the host object in the db and the display name.) Calling this method will return the first item if it is an array. For non-array values the item is returned unmodified. Example: miq_provision_request.get_option(:number_of_cpus)
get_tag(category)	Returns the tags selected for the specified tag category. Example: request.get_tag(:department) Returns: ["accounting", "engineering"]
get_tags	Get all selected tags stored in a hash by category. If more than one tag is selected in a category, the hash will contain an array of tag names. Otherwise it will contain the tag name as a string. Example: request.get_tags Returns: {:cc⇒"001", :department⇒["accounting", "engineering"]}
miq_request	(Legacy support) Internal Note: The miq_request instance use to be a separate instance from the specific request instance (like miq_provision_request). When the classes were refactored into 1 this method was added to allow existing code and automate methods to continue to run unchanged.)
miq_request_tasks	Returns the requests tasks
options	Returns a hash containing all the options set for the current provision object
pending	Puts the object in a pending state for approval Example: # Raise automation event: request_pending $evm.root["miq_request"].pending
reason	Returns reason for approval or denial of request
requester	Returns the requester
resource	Returns the resource for the request
set_message(value)	Sets the message for the request

Method	Use
set_option(key, value)	Sets the specified key/value pair for the object

Automation Request (automation_request)

Method	Use
automation_tasks	Returns objects automate tasks

Host Provision Request (miq_host_provision_request)

Method	Use
miq_host_provisions	Returns the miq_host_provisions objects
ci_type	Returns the cloud infrastructure type: host

VM Provision Request (miq_provision_request)

Method	Use
check_quota(quota_type, options={})	Returns the quota information for the specified type
ci_type	Returns the cloud infrastructure type: vm
eligible_clusters	Returns an array of available Cluster objects filtered by previously selected resources
eligible_customization_templates	Returns an array of available Customization Templates filtered by previously selected resources
eligible_folders	Returns an array of available Folder objects filtered by previously selected resources
eligible_hosts	Returns an array of available Host objects filtered by previously selected resources
eligible_iso_images	Returns an array of available ISO image objects filtered by previously selected resources
eligible_pxe_images	Returns an array of available PXE Image objects filtered by previously selected resources
eligible_pxe_servers	Returns an array of available PXE Server objects filtered by previously selected resources
eligible_resource_pools	Returns an array of available Resource Pool objects filtered by previously selected resources
eligible_resources(rsc_type)	Returns eligible resources given the type specified
eligible_storages	Returns an array of available Storage (Datastore) objects filtered by previously selected resources
eligible_windows_images	Returns an array of available Windows Image objects filtered by previously selected resources
get_folder_paths	Returns a hash where the key is an index and the value is the fully-qualified path name of the folder. (Sample: {7 ⇒ Dev/Dept1/QA, 8 ⇒ Test/Dept2/QA}) This format is useful when a fully-qualified path is required to match the folder name. For example, if you had multiple QA folders under different departments in the sample above. To find the proper QA folder you need to evaluate the entire folder path.
get_retirement_days	Returns the number of dates until retirement
miq_provision	Returns the task.
miq_request	Returns the miq_provision_requests miq_request object
set_cluster(rsc)	Set the cluster to use based on object returned from eligible_clusters
set_customization_template(rsc)	Set the customization_template to use based on object returned from eligible_customization_templates
set_folder(folder_path)	Set the folder to use based on object returned from eligible_folders. In addition, set_folder accepts the following folder types: Folder Paths - separated by forward slashes. (Must include Data-center name.) Example: Prod/Discovered virtual machine where Prod is the Data-center name and Discovered virtual machine is the folder name. Object returned from the get_folder_paths method
set_host(rsc)	Set the host to use based on object returned from eligible_hosts

Method	Use
set_network_adapter(idx, nic_hash, value=nil)	Modifies the network card attached to the VM container Available settings: :is_dvs true / false (Default: false) :network (Network Name) :mac_address :devicetype (Default: VirtualPCNet32) Defined by VMware: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/vim.vm.device.VirtualEthernetCard.html :connectable ⇒ {:allowguestcontrol ⇒ true / false} (Default: true) :connectable ⇒ {:startconnected ⇒ true / false} (Default: true) :connectable ⇒ {:connected ⇒ true / false} (Default: true) Example: prov.set_network_adapter(1, {:network ⇒ dvs_net1, :is_dvs ⇒ true})
set_network_address_mode(mode)	Sets IP address type. Available modes are dhcp and static
set_nic_settings(idx, nic_hash, value=nil)	Modifies the network interface settings at the operating system level Available settings: :addr_mode "dhcp" / "static" (Default: static) :ip_addr :subnet_mask :gateway :dns_domain :dns_servers (Windows Only) Comma separated values :sysprep_netbios_mode (Windows Only) Defined by VMware: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/vim.vm.customization.IPSettings.NetBIOSMode.html :wins_servers Passed as a string specifying the Primary and Secondary WINS servers separated by a comma. "<PrimaryWINS>, <SecondaryWINS>" Example: prov.set_nic_settings(1, {:ip_addr⇒10.226.133.55, :subnet_mask⇒'255.255.255.192', :gateway⇒'10.226.133.5', :addr_mode⇒["static", "Static"] })
set_iso_image(rsc)	Set the iso_image to use based on object returned from eligible_iso_images
set_pxe_image(rsc)	Set the pxe_image to use based on object returned from eligible_pxe_images
set_pxe_server(rsc)	Set the pxe_server to use based on object returned from eligible_pxe_servers
set_resource_pool(rsc)	Set the resource_pool to use based on object returned from eligible_resource_pools
set_resource(rsc)	Sets the resource for the request. (Helper method, should not be called directly)
set_retirement_days	Set the number of days until retirement.
set_storage(rsc)	Set the Datastore (storage object) to use based on object returned from eligible_storages
set_vm_notes(note)	Sets text for the VM notes (aka annotation) field
set_windows_image(rsc)	Set the windows_image to use based on object returned from eligible_windows_images
source_type	Returns the provision source type. (values are 'vm' or 'template')
src_vm_id	Returns ID of the template being cloned
target_type	Returns the provision target type. (values are 'vm' or 'template')

Method	Use
vm_template	Returns the requests template

Request Task (miq_request_task)

Method	Use
add_tag(category, tag_name)	Add a tag to a provision instance by specifying the category and tag name. Example: miq_provision.add_tag(:location, "CHI")
clear_tag(category=nil, tag_name=nil)	Without any parameters, the clear_tag method will clear all tags from the provision request. Providing a category will clear all tags selected in that category. Clear a specific category/tag by providing it. Example: miq_provision.clear_tag(:location, "CHI")
destination	Returns the destination object. (The resultant object from running the task. In the case of provisioning, this would be the newly created VM.)
execute	Executes or processes the request.
finished(msg)	Sets the task to finished with the supplied message.
get_classification(category)	Works the same as get_tag(category) but the returned data is a hash with :name and :description. Example: request.get_classification(:department) Returns: [{:name⇒"accounting", :description⇒"Accounting"}, {:name⇒"engineering", :description⇒"Engineering"}]
get_classifications	Works the same as get_tag but the returned tag data is a hash with :name and :description. Example: request.get_classifications Returns: {:cc⇒{:name⇒"001", :description⇒"Cost Center 001"}, :department⇒[{:name⇒"accounting", :description⇒"Accounting"}, {:name⇒"engineering", :description⇒"Engineering"}]}
get_option_last(key)	This method is the same as get_option, except that it returns the last array value.
get_option(key)	Returns a value from the options hash based on the name of the key name passed in. Internally many of the values are stored as an array of items. (For example, a target host would be stored as the index to the host object in the db and the display name.) Calling this method will return the first item if it is an array. For non-array values the item is returned unmodified. Example: miq_provision_request.get_option(:number_of_cpus)
get_tag(category)	Returns the tags selected for the specified tag category. Example: request.get_tag(:department) Returns: ["accounting", "engineering"]
get_tags	Get all selected tags stored in a hash by category. If more than one tag is selected in a category, the hash will contain an array of tag names. Otherwise it will contain the tag name as a string. Example: request.get_tag Returns: {:cc⇒"001", :department⇒["accounting", "engineering"]}
message=(msg)	Sets the message for the request task.
miq_request	Returns the miq_request for the task.
miq_request_task	Returns the parent miq_request task.
miq_request.user_message=(msg)	Allows automate method to override the default request status message. As a result, you can set custom messages for the progression of the state machine and its success or failure. This message will appear as the last message for the service request.
miq_request_tasks	Returns the children miq_request tasks.
options	Returns a hash containing all the options set for the current object.
set_option(key, value)	Updates a key/value pair in the options hash for the provision object. Often the value is required to be an array.
source	Returns the source object. (The source, or input, object that the task runs against. In the case of provisioning, this would be the VM or template selected to be provisioned.)

Automation Task (automation_task)

Method	Use

Method	Use
automation_request	Returns associated automation_request object
status	Returns status of the task

Host Provision Task (miq_host_provision)

Method	Use
host	Returns objects host
miq_host_provision_request	Returns the request that created the task
status	Returns status of host provision

VM Provision Task (miq_provision)

Method	Use
check_quota(quota_type, options={})	Returns the quota information for the specified type
eligible_clusters	Returns an array of available Cluster objects filtered by previously selected resources
eligible_customization_templates	Returns an array of available Customization Templates
eligible_folders	Returns an array of available Folder objects filtered by previously selected resources
eligible_hosts	Returns an array of available Host objects filtered by previously selected resources
eligible_iso_images	Returns an array of available ISO Image objects filtered by previously selected resources
eligible_pxe_images	Returns an array of available PXE Image objects filtered by previously selected resources
eligible_pxe_servers	Returns an array of available PXE Servers filtered by previously selected resources
eligible_resource_pools	Returns an array of available Resource Pool objects filtered by previously selected resources
eligible_resources(rsc_type)	Returns the eligible resources for the resource type specified
eligible_storages	Returns an array of available Storage (Datastore) objects filtered by previously selected resources
eligible_windows_images	Returns an array of available Windows Image objects filtered by previously selected resources
get_domain_details	Returns domain information
get_domain_name	Returns domain name
get_folder_paths	Returns a hash where the key is an index and the value is the fully-qualified path name of the folder. (Sample: {7 ⇒ Dev/Dept1/QA, 8 ⇒ Test/Dept2/QA}) This format is useful when a fully-qualified path is required to match the folder name. For example, if you had multiple QA folders under different departments in the sample above. To find the proper QA folder you need to evaluate the entire folder path.
get_network_details	Returns network information
get_network_scope	Returns network scope
miq_provision_request	Returns the provision request object
set_cluster(rsc)	Set the cluster to use based on object returned from eligible_clusters
set_customization_spec(name=nil, override=false)	Sets the name of the custom spec to use as defined by its name in Virtual Center (Note: This is not supported when being passed by VMware.)
set_customization_template(rsc)	Set the customization_template to use based on object returned from eligible_customization_templates
set_dvs(portgroup, switch = portgroup)	Set the name of the Distributed Virtual Switch (portgroup). An options <switch> name can also be passed Example: miq_provision.set_dvs('default')

Method	Use
set_folder(folder_path)	Set the folder to use based on object returned from eligible_folders. In addition, set_folder accepts the following folder types: Folder Paths - separated by forward slashes. (Must include Data-center name.) Example: Prod/Discovered virtual machine where Prod is the Data-center name and Discovered virtual machine is the folder name. Object returned from the get_folder_paths method
set_host(rsc)	Set the host to use based on object returned from eligible_hosts
set_network_adapter(idx, nic_hash, value=nil)	Modifies the network card attached to the VM container Available settings: :is_dvs true / false (Default: false) :network (Network Name) :mac_address :devicetype (Default: VirtualPCNet32) Defined by VMware: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/vim.vm.device.VirtualEthernetCard.html :connectable ⇒ {:allowguestcontrol ⇒ true / false} (Default: true) :connectable ⇒ {:startconnected ⇒ true / false} (Default: true) :connectable ⇒ {:connected ⇒ true / false} (Default: true) Example: prov.set_network_adapter(1, {:network ⇒ dvs_net1, :is_dvs ⇒ true})
set_network_address_mode(mode)	Available modes are dhcp and static
set_nic_settings(idx, nic_hash, value=nil)	Modifies the network interface settings at the operating system level Available settings: :addr_mode "dhcp" / "static" (Default: Statis) :ip_addr :subnet_mask :gateway :dns_domain :dns_servers (Windows Only) Comma separated values :sysprep_netbios_mode (Windows Only) Defined by VMware: http://www.vmware.com/support/developer/vc-sdk/visdk400pubs/ReferenceGuide/vim.vm.customization.IPSettings.NetBIOSMode.html :wins_servers Passed as a string specifying the Primary and Secondary WINS servers separated by a comma. "<PrimaryWINS>, <SecondaryWINS>" Example: prov.set_nic_settings(1, {:ip_addr⇒10.226.133.55, :subnet_mask⇒'255.255.255.192', :gateway⇒'10.226.133.5', :addr_mode⇒["static", "Static"] })
set_iso_image(rsc)	Set the iso_image to use based on object returned from eligible_iso_images
set_pxe_image(rsc)	Set the pxe_image to use based on object returned from eligible_pxe_images
set_pxe_server(rsc)	Set the pxe_server to use based on object returned from eligible_pxe_servers
set_resource_pool(rsc)	Set the resource_pool to use based on object returned from eligible_resource_pools
set_storage(rsc)	Set the Datastore (storage object) to use based on object returned from eligible_storages
set_vlan(vlan)	Sets the name of the VLan to use Example: miq_provision.set_vlan('default')
set_vm_notes(note)	Sets text for the VM notes (aka annotation) field

Method	Use
set_vm_notes(notes)	Sets text for the VM notes (aka annotation) field
set_windows_image(rsc)	Set the windows_image to use based on object returned from eligible_windows_images
source_type	Returns the provision source type. (values are 'vm' or 'template')
status	Returns provision status
target_type	Returns the provision target type. (values are 'vm' or 'template')
vdi_farm	Returns VDI Farm information
vm	The newly created vm
vm_template	Returns the template selected to be provisioned

```
begin
  miq_provision = $evm.root["miq_provision"] || $evm.root['miq_provision']
  prov = $evm.root["miq_provision"]
  user = prov.miq_request.requester
  raise "User not specified" if user.nil?



  ###################################
  # Process Change Request Number and set VM Annotation
  ###################################
  intake = prov.get_option(:vm_description)
  intake = "Change Request#: #{intake}"
  prov.set_option(:vm_description,intake)


  ###################################
  # Set the customization spec based on the environment tag chosen in the dialog
  ###################################
  tags = prov.get_tags
  $evm.log("info","Tags: #{tags.inspect}")
  env = tags[:environment]
  $evm.log("info", "Mapping custom spec based on Category Environment <#{env}> chosen in the dialog")
  if env.eql? "dev"
    customization_spec = "Dev-Specification"
    miq_provision.set_customization_spec(customization_spec)
  end
  if env.eql? "stg"
    customization_spec = "Stg-Specification"
    miq_provision.set_customization_spec(customization_spec)
  end


  ###################################
  # Set the VM Notes as follows:
  ###################################
  vm_notes = "#{intake}"
  vm_notes +=  "\nOwner: #{miq_provision.get_option(:owner_first_name)} #
{miq_provision.get_option(:owner_last_name)}"
  vm_notes += "\nEmail: #{miq_provision.get_option(:owner_email)}"
  vm_notes += "\nSource Template: #{miq_provision.vm_template.name}"
  miq_provision.set_vm_notes(vm_notes)


  ###################################
  # Drop the VM in the targeted folder
  # In VC a folder must exist that matches the LDAP Group
  # VM will be placed in the Folder
  ###################################
  if prov.get_option(:placement_folder_name).nil?
    ###################################
    # If you want to use a Default folder, set folder = below to the default
    ###################################
    #    folder = "22F DC/LAB FARM/GSE/Intel/Infrastructure/ManageIQ/SelfServiceVMs"
    folder = "DC1/Infrastructure/ManageIQ/SelfService"
    $evm.log("info", "Placing VM in VC folder: <#{folder}")
    $evm.log("info", "Set_folder called with [#{folder.inspect}]")

    miq_provision.set_folder(folder)
  end


  ###################################################
  # Set the IP Address based on the :mac_address entered in the dialog
  #
```

```
  ###################################################
  ipaddr = prov.get_option(:mac_address)

  if ! ipaddr.nil?
    # Set provisioning options to override options
    prov.set_option(:sysprep_spec_override, [true, 1])
    prov.set_option(:addr_mode, ["static", "Static"])
    prov.set_option(:ip_addr, ipaddr)
    # Reset :mac_address to nil
    prov.set_option(:mac_address, nil)
  end

  $evm.log("info", "Provision Options: #{prov.options.inspect}")

  exit MIQ_OK

rescue => err
  $evm.log("info", "Set_folder err [#{err}]\n#{err.backtrace.join("\n")}")
end
```

Adding Disks to OpenStack Instance Provisioning

Using the clone_options, construct the keys and values required to define new disks as well as override or set other keys if needed.

Include the following settings with prov.set_options to set one or multiple volumes to attach. Note that each hash requires the :volume_id and :device_name keys.

```
clone_options = {
  :block_device_mapping => [
    {:volume_id => "d4xxxx92-5xx0-4xx9-bxx4-f1xxxxxxx19", :device_name => "/dev/sdb"},
    {:volume_id => "e7xxxxf5-dxxd-4xx0-8xx3-a2xxxxxxx12", :device_name => "/dev/sdc"},
  ]
}

prov = $evm.root["miq_provision"]
prov.set_option(:clone_options, clone_options)
```

Booting OpenStack Instances from Volume

You can boot instances created in Red Hat Enterprise Linux OpenStack Platform from a specified volume from an existing selection of block volumes. You can enable automate to pass the required parameters to OpenStack to use volumes as booting devices on an instance.

Include the following settings in the prov.set_option for provisioning instances created in OpenStack.

```
prov.set_option(
  :clone_options, {
    :image_ref => nil,
    :block_device_mapping_v2 => [{
      :boot_index => 0,
      :uuid => "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx",
      :device_name => "vda",
      :source_type => "volume",
      :destination_type => "volume",
      :delete_on_termination => false
    }]
  }
)
```

You can allow passing “block_device_mapping_v2” in the clone options and remove keys from clone_options by setting value to nil.

VM Provisioning for Clouds Task (miq_provision_cloud)

Method	Use
availability_zones	Returns object’s availability zones
instance_types	Returns object’s instance types
security_groups	Returns object’s security groups
floating_ip_addresses	Returns object’s floating IP addresses
cloud_networks	Returns object’s cloud network
cloud_subnets	Returns object’s cloud subnet
guest_access_key_pairs	Returns object’s guest key pair
cloud_tenants	Returns object’s cloud tenant

Service Template Provision Task (service_template_provision_task)

Method	Use

Method	Use
dialog_options	Returns objects dialog options hash
dialog_parser	Provides consolidated and configurable dialog parsing
get_dialog_option(key)	Returns objects dialog value for the specified key
service_resource	Returns the service resource for the task
set_dialog_option(key, value)	Sets a dialog option Example: set_dialog_option('memory',memory_size)
status	Returns the tasks status

Service Reconfiguration Task (service_reconfigure_task)

Method	Use
dialog_options	Show all dialog options for object
get_dialog_option(key)	Show a dialog option based stored in key
set_dialog_option(key, value)	Set value as a dialog option in key
status	Returns status of the task
finished(msg)	Sets the task to finished with the supplied message

VM Migrate Task (vm_migrate_task)

Method	Use
status	Returns status of the migration task

Proxies (miq_proxy)

Method	Use
host	Returns object's hosts
powershell(script, returns = 'string')	Submits a powershell script

Servers (miq_server)

These methods are available to the Infrastructure Automation Server.

Method	Use
zone	Returns Infrastructure Automation Servers Zone
region_number	Returns Infrastructure Automation Servers Region Number
region_name	Returns Infrastructure Automation Servers Region Name

```
  ##########################
  #
  # Method: dumpServer
  # Inputs: $evm.root['miq_server']
  # Description: Dump MIQ Server information
  #
  ##########################
  def dumpServer
    $evm.log("info","#{@method} - Server:<#{$evm.root['miq_server'].name}> Begin Attributes")
    $evm.root['miq_server'].attributes.sort.each { |k, v| $evm.log("info", "#{@method} - Server:<#
{$evm.root['miq_server'].name}> Attributes - #{k}: #{v.inspect}")}
    $evm.log("info","#{@method} - Server:<#{$evm.root['miq_server'].name}> End Attributes")
    $evm.log("info", "")
  end
```

Network (network)

Method	Use
hardware	Returns objects hardware
guest_device	Returns objects guest devices

PXE Image (pxe_image)

Method	Use
customization_templates	Returns objects customization templates
pxe_server	Returns objects pxe server

PXE Server (pxe_server)

Method	Use
advertised_images	Returns objects advertised images
advertised_pxe_images	Returns objects advertised pxe images
default_pxe_image_for_windows	Returns objects default pxe image for windows
discovered_images	Returns objects discovered images
discovered_pxe_images	Returns objects discovered pxe images
images	Returns objects images
pxe_images	Returns objects pxe_images
windows_images	Returns objects windows images

Security Groups (security_group)

Method	Use
ext_management_system	Returns object's Management System
cloud_network	Returns object's cloud network
cloud_tenant	Returns object's cloud tenant
firewall_rules	Returns object's firewall rules
vms	Returns object's VMs

Service (service)

Method	Use
custom_keys	Returns custom keys
custom_get(key)	Gets value for specified custom key
custom_set(attribute, value)	Sets value for specified custom key
display=(display)	Set display option
group=(group)	Sets group that owns the service
name=(new_name)	Sets name of service
owner=(owner)	Sets owner of the service
request_retire	Retire Service immediately
retirement_warn=(seconds)	Sets when to send retirement warning
retires_on=(date)	Sets retirement date
shutdown_guest	Shuts downs guest operating system of the Service
start	Start the Service
stop	Stop the Service
suspend	Suspend the Service
vms	Show all virtual machines associated with this service
direct_vms	Show virtual machines directly associated with this service
indirect_vms	Show virtual machines associated with lower level services in the hierarchy
root_service	Show the top level service in the hierarchy for the target service
all_service_children	Show all lower level services to the target service in the hierarchy
direct_service_children	Show direct services associated with the target service
indirect_service_children	Show services associated with lower level services of the target service
parent_service	Show the parent service for the target service
description=(new_description)	Sets the service description
remove_from_vmdb	Delete the service from the database
dialog_options	Returns all dialog options
get_dialog_option(key)	Returns a specific dialog option specified by key
set_dialog_option(key, value)	Sets value of a dialog option specified by key

Service Resource (service_resource)

Method	Use
service	Returns the associated service
service_template	Returns the associated service template
resource	Returns the resource for the request
source	Returns the source object

Service Template (service_template)

Method	Use
group=(group)	Sets group for the service template
owner=(owner)	Sets owner for the service template

Snapshot (snapshot)

These methods can be used on Snapshots

Method	Use
vm	Returns Snapshots VM
current?	Checks to see if this is the current snapshot
get_current_snapshot	Returns the current snapshot id
revert_to	Reverts to specified snapshot
remove	Removes specified snapshot

Datastores (storage)

Method	Use
ext_management_systems,	Returns objects Management System
hosts	Returns objects Hosts
vms	Returns objects Virtual Machines
unregistered_vms	Returns objects unregistered Virtual Machines
to_s	Converts object to string
scan	Performs SmartState Analysis on the object

```
 ##########################
  #
  # Method: dumpStorage
  # Inputs: $evm.root['storage']
  # Description: Dump Storage information
  #
  ##########################
  def dumpStorage(storage)
    $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> Begin Attributes")
    storage.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - Storage:<#{storage.name}>
Attributes - #{k}: #{v.inspect}")}
    $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> End Attributes")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> Begin Associations")
    storage.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix} - Storage:<#{storage.name}>
Associations - #{assc}")}
    $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> End Associations")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> Begin Virtual Columns")
    storage.virtual_column_names.sort.each { |vcn| $evm.log("info", "#{@log_prefix} - Storage:<#
{storage.name}> Virtual Columns - #{vcn}: #{storage.send(vcn)}")}
    $evm.log("info","#{@log_prefix} - Storage:<#{storage.name}> End Virtual Columns")
    $evm.log("info","")
  end
```

Switch (switch)

These methods can be used on Switches.

Method	Use
host	Returns switch’s Host
guest_devices	Returns switch’s guest devices
lans	Returns switch’s lans

User (user)

These methods can be used on the currently logged on user.

Method	Use
current_group	Returns user’s assigned internal group
custom_get(key)	Returns the custom key value specified by “key”
custom_keys	Returns an array of custom keys
custom_set(key,value)	Sets custom value for “key” to “value”

Method	Use
email	Returns user's email address
get_ladap_attribute(name)	Returns the value of the specified LDAP attribute
get_ldap_atttribute_names	Returns user's LDAP attribute names
ldap_group	Returns user's assigned LDAP group
miq_requests	Returns user's requests
name	Returns user's name
userid	Returns user's userid
vms	Returns Virtual Machines that this user owns

```
  ##########################
  #
  # Method: dumpUser
  # Inputs: $evm.root['user']
  # Description: Dump User information
  #
  ##########################
  def dumpUser
    user = $evm.root['user']
    unless user.nil?
      $evm.log("info","#{@method} - User:<#{user.name}> Begin Attributes [user.attributes]")
      user.attributes.sort.each { |k, v| $evm.log("info", "#{@method} - User:<#{user.name}> Attributes - #
{k}: #{v.inspect}")}
      $evm.log("info","#{@method} - User:<#{user.name}> End Attributes [user.attributes]")
      $evm.log("info", "")

      $evm.log("info","#{@method} - User:<#{user.name}> Begin Associations [user.associations]")
      user.associations.sort.each { |assc| $evm.log("info", "#{@method} - User:<#{user.name}> Associations -
#{assc}")}
      $evm.log("info","#{@method} - User:<#{user.name}> End Associations [user.associations]")
      $evm.log("info","")

      $evm.log("info","#{@method} - User:<#{user.name}> Begin Virtual Columns [user.virtual_column_names]")
      user.virtual_column_names.sort.each { |vcn| $evm.log("info", "#{@method} - User:<#{user.name}> Virtual
Columns - #{vcn}: #{user.send(vcn).inspect}")}
      $evm.log("info","#{@method} - User:<#{user.name}> End Virtual Columns [user.virtual_column_names]")
      $evm.log("info","")
    end
  end
```

Virtual Machines and Templates (vm_or_template)

The following methods can be used on a virtual machine or template object.

Method	Use
add_disk [1]	Add storage to an existing VM. This method accepts as parameters: disk_name, disk_size_mb, and options to pass values for :thinProvisioned, :dependent, and :persistent. Example: `vm.add_disk("[#{vm.storage_name}]", size_in_mb, thin_provisioned: true)`
changed_vm_value?	Checks the 2 most recent drift state captures , and answers whether the specified value changed between them
collect_running_processes	Collects the running processes of the object
create_snapshot(name, desc = nil)	Create a snapshot of the object. The **name** parameter is not used for Red Hat Virtualization virtual machines when creating a snapshot.
custom_get(key)	Get the value of specified Infrastructure Automation Server key from the object
custom_keys	List all Infrastructure Automation Server custom keys for the object
custom_set(key, value)	Set a custom Infrastructure Automation Server key value
datacenter	Returns objects Datacenter
direct_service	Show the direct service relationship of the virtual machine
directories	Returns number of directories on the object
ems_blue_folder (this will be reworked to be more VMware-specific)	Returns objects blue folder from VMware. These are the folders showing in VM and Templates view in VMware
ems_cluster	Returns objects cluster
ems_custom_get(key)	Gets specified key of custom Management System Attribute
ems_custom_keys	List the custom keys defined by the Management System for the object
ems_custom_set(attribute, value)	Sets specified key and value of custom Management System Attribute
ems_folder	Returns objects folder on Management System

Method	Use
ems_ref_string	Returns unique identifier the Management System uses to identify this resource. For example, in VMware a VM would return a value like: "vm-26622"
event_log_threshold?(options)	Searches event log records to determine if an event has occurred x number of times within a defined time frame. Returns true if the number of matching records found are greater or equal to the specified freq_threshold, otherwise it returns false Options values: :message_filter_type - Must be one of "STARTS WITH", "ENDS WITH", "INCLUDES", "REGULAR EXPRESSION" :message_filter_value - <string value to search for> :time_threshold - Options time interval to search. Example: 2.days (Search the past 2 days of event logs) Default 10.days :freq_threshold - Number of occurrences to check for. Default = 2 :source, :event_id, :level, :name - Options filter values
event_threshold?(options)	Checks if an event (or multiple events) have occurred X number of times in N seconds. The values below are used if no data is passed event_threshold?(options = {:time_threshold ⇒ 30.minutes, :event_types ⇒ ["MigrateVM_Task_Complete"], :freq_threshold ⇒ 2})
ext_management_system	Returns objects Management System
files	Returns number of files on the object
get_realtime_metric(metric, range, function)	Returns specified realtime metric
group=(group)	Sets objects group
guest_applications	Returns objects Guest Application list
hardware	Returns objects Hardware
host	Returns objects Host
migrate(host, pool = nil, priority = "defaultPriority", state = nil)	Migrates the object to another host. The only required parameter is host
miq_provision	If VM was created using Infrastructure Automation Server provisioning, this is the miq_provision task instance that created the VM
move_into_folder(folder) [2]	Use this method to move VMware virtual machines into folders. Example: vm = Vm.find_by_name('<vm_name>') folder = EmsFolder.find_by(:name ⇒ '<folder_name>') vm.move_into_folder(folder) unless folder.nil?
operating_system	Returns objects Operating System
owner	Return objects owner
owner=(owner)	Sets objects owner
performances_maintains_value_for_duration?	Based on options given, checks to see if a performance threshold is maintained for a time period Example: vm.performances_maintains_value_for_duration?(:column ⇒ "cpu_usage_rate_average", :operator ⇒ "=", :value ⇒ 3.51, :duration ⇒ 20.minutes)
reboot_guest	Reboots the guest operating system
reconfigured_hardware_value?	Checks if hardware value has been reconfigured
refresh	Refresh power states and relationships of the object
registered?	Is the object registered?
remove_all_snapshots	Remove all of the objects snapshots
remove_from_disk	Removes the object from disk
remove_from_vmdb	Removes the object from the VMDB
remove_snapshot(snapshot_id)	Remove a specific snapshot based on the snapshot_id
resource_pool	Returns objects Resource Pool
retire_now	Retire the object immediately
retirement_warn=(seconds)	Send a retirement warning

Method	Use
retires_on=(date)	Retire the object on date specified
revert_to_snapshot(snapshot_id)	Revert to a snapshot based on the snapshot_id
scan(scan_categories = nil)	Perform SmartState Analysis on the object. Scan_categories is optional
service	Show the top-level service for a virtual machine in a service hierarchy. For the immediate parent service relationship of a virtual machine, use direct_service
shutdown_guest	Shuts down the guest operating system of the object
snapshots	Returns list of snapshots for the object
standby_guest	Puts the operating system on standby
start	Starts the object See Samples/PowerOn_DHOB
stop	Stops the object
storage	Returns objects Datastore
suspend	Suspends the object
to_s	Converts object to string
unlink_storage	Removes the reference to the VM's Datastore
unregister	Unregisters the object from the Management System

```
  ############################
  #
  # Method: dumpVM
  # Inputs: $evm.root['vm']
  # Description: Dump VM information
  #
  ############################
  def dumpVM(vm)
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Attributes [vm.attributes]")
    vm.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> Attributes - #{k}: #
{v.inspect}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Attributes [vm.attributes]")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Associations [vm.associations]")
    vm.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> Associations - #
{assc}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Associations [vm.associations]")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Hardware Attributes [vm.hardware]")
    vm.hardware.attributes.each { |k,v| $evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> Hardware - #{k}: #
{v.inspect}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Hardware Attributes [vm.hardware]")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Hardware Associations
[vm.hardware.associations]")
    vm.hardware.associations.sort.each { |assc| $evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> hardware
Associations - #{assc}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End hardware Associations [vm.hardware.associations]")
    $evm.log("info","")

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Neworks [vm.hardware.nics]")
    vm.hardware.nics.each { |nic| nic.attributes.sort.each { |k,v| $evm.log("info", "#{@log_prefix} - VM:<#
{vm.name}> VLAN:<#{nic.device_name}> - #{k}: #{v.inspect}")}}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Networks [vm.hardware.nics]")
    $evm.log("info","")

    unless vm.ext_management_system.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin EMS [vm.ext_management_system]")
      vm.ext_management_system.attributes.sort.each { |ems_k, ems_v| $evm.log("info", "#{@log_prefix} - VM:<#
{vm.name}> EMS:<#{vm.ext_management_system.name}> #{ems_k} - #{ems_v.inspect}")}
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End EMS [vm.ext_management_system]")
      $evm.log("info","")
    end

    unless vm.owner.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Owner [vm.owner]")
      vm.owner.attributes.each { |k,v| $evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> Owner - #{k}: #
{v.inspect}")}
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Owner [vm.owner]")
      $evm.log("info","")
    end
```

```
    unless vm.operating_system.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Operating System [vm.operating_system]")
      vm.operating_system.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - VM:<#{vm.name}>
Operating System - #{k}: #{v.inspect}")}
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Operating System [vm.operating_system]")
      $evm.log("info","")
    end

    unless vm.guest_applications.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Guest Applications")
      vm.guest_applications.each { |guest_app| guest_app.attributes.sort.each { |k, v| $evm.log("info", "#
{@log_prefix} - VM:<#{vm.name}> Guest Application:<#{guest_app.name}> - #{k}: #{v.inspect}")}} unless
vm.guest_applications.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Guest Applications")
      $evm.log("info","")
    end

    unless vm.snapshots.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Snapshots")
      vm.snapshots.each { |ss| ss.attributes.sort.each { |k, v| $evm.log("info", "#{@log_prefix} - VM:<#
{vm.name}> Snapshot:<#{ss.name}> - #{k}: #{v.inspect}")}} unless vm.snapshots.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Snapshots")
      $evm.log("info","")
    end

    unless vm.storage.nil?
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin VM Storage [vm.storage]")
      vm.storage.attributes.sort.each { |stor_k, stor_v| $evm.log("info", "#{@log_prefix} - VM:<#{vm.name}>
Storage:<#{vm.storage.name}> #{stor_k} - #{stor_v.inspect}")}
      $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End VM Storage [vm.storage]")
      $evm.log("info","")
    end

    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> Begin Virtual Columns -----")
    vm.virtual_column_names.sort.each { |vcn| $evm.log("info", "#{@log_prefix} - VM:<#{vm.name}> Virtual
Columns - #{vcn}: #{vm.send(vcn).inspect}")}
    $evm.log("info","#{@log_prefix} - VM:<#{vm.name}> End Virtual Columns -----")
    $evm.log("info","")
  end

  ####################
  #
  # Method: createSnapshot
  #
  ####################

  def createSnapshot(vm, snap_name, snap_desc=snap_name)
    $evm.log("info","#{@method} - VM:<#{vm.name}> Creating Snapshot:<#{snap_name}> Description:<#
{snap_desc}>")
    vm.create_snapshot(snap_name, snap_desc)
  end

  ####################
  # Red Hat Virtualization Virtual Machines
  # Method: createSnapshot
  #
  ####################

    def createSnapshot(vm, snap_desc=snap_name)
        $evm.log("info","#{@method} - VM:<#{vm.name}> Creating Snapshot:<#{snap_name}> Description:<#
{snap_desc}>")
        vm.create_snapshot(snap_desc)
    end
```

VMs (vm)

Method	Use
add_to_service(service)	Adds the VM to a service object
remove_from_service	Removes the VM from its parent service

The following table lists the metric types available for the **`get_realtime_metric(metric, range, function)`** method for virtual machines.

Metric	Description
v_derived_storage_used	Capacity - Used space in bytes
v_pct_cpu_ready_delta_summation	CPU - Percentage ready
v_pct_cpu_wait_delta_summation	CPU - Percentage wait
v_pct_cpu_used_delta_summation	CPU - Percentage used
v_derived_vm_count	State - Peak average virtual machines (Hourly Count / Daily Average)

Metric	Description
v_derived_cpu_reserved_pct	CPU - Percentage available
v_derived_memory_reserved_pct	Memory - Percentage available

The following Ruby snippet demonstrates using the **`get_realtime_metric(metric, range, function)`** method using the **`v_pct_cpu_ready_delta_summation`** metric.

```
host = $evm.root['vm']
cpu_rdy = vm.get_realtime_metric(:v_pct_cpu_ready_delta_summation, [15.minutes.ago.utc,5.minutes.ago.utc],
:avg)
```

VMs for Clouds (vm_cloud)

Method	Use
availability_zone	Returns object's availability zone
flavor	Returns object's favor
cloud_network	Returns object's cloud network
cloud_subnet	Returns object's cloud subnet
floating_ip	Returns object's floating IPs
security_groups	Returns object's security groups
key_pairs	Returns object's key pairs

Windows Image (windows_image)

Method	Use
customization_templates	Returns the images customization templates.
pxe_server	Returns the images pxe server.

Creating Categories and Tags

These methods are invoked using $evm.execute. See after the table for usage examples.

Method(parameter)	Use
category_exists?	Checks to see if a tag category exists. Usually used with unless or if to then create a category that does not exist. Example: $evm.execute('category_exists?', "department")
category_create	Creates a new category, and if only one tag value from this category can be assigned to a configuration item, sets that tag value using the **`single_value`** option. Example: $evm.execute('category_create', :name ⇒ "department", :single_value ⇒ false, :description ⇒ "Department") **NOTE:** The **`category_create`** method currently does not work as expected. The user interface shows the value provided for **`:description`** in the **Display Name** field and the value provided for **`:example_text`** is displayed in the **Description** field.
tag_exists?	Checks to see if a tag exists in a category. Example: $evm.execute('tag_exists?', "department", "finance")
tag_create	Creates a new tag in the specified category. Example: $evm.execute('tag_create', "department", :name ⇒ "finance", :description ⇒ "Finance")

Category and Tags Methods Example

In this example, the VMDB is checked to see if the **Department** category exists. If it does, then a message is logged. If not, the category is created and a message is logged. Values are then added to the category.

```
if $evm.execute('category_exists?', "department")
  $evm.log("info", "Classification department exists")
else
  $evm.log("info", "Classification department doesn't exist, creating category")
  $evm.execute('category_create', :name => "department", :single_value => false, :description =>
"Department")
  $evm.log("info", "Adding new tag in Department Category")
  $evm.execute('tag_create', "department", :name => "finance", :description => "Finance")
```

Quota

Method(parameter)	Use

Method(parameter)	Use
vms_by_owner, vms_by_group, vms_by_owner_and_group	Collect stats about existing VMs by owner or in the same LDAP group as the owner in the current request. Sample return object: {:class_name⇒"Vm", :count⇒5, :ids⇒[22, 120, 121, 122, 117], :cpu⇒6, :memory⇒5120, :allocated_storage⇒29032972288, :used_storage⇒29032972288}
retired_vms_by_owner, retired_vms_by_group, retired_vms_by_owner_and_group	Collect stats about retired VMs, that have not been deleted from the host, by owner or in the same LDAP group as the owner in the current request. Sample return object: {:class_name⇒"Vm", :count⇒5, :ids⇒[22, 120, 121, 122, 117], :cpu⇒6, :memory⇒5120, :allocated_storage⇒29032972288, :used_storage⇒29032972288}
provisions_by_owner, provisions_by_group	Collect stats based on provisions running on the same day as the current request (based on scheduled date or immediate) by owner or in the same LDAP group as the owner. (Results do not include Provision Requests that have not been approved.) Sample return object: {:class_name⇒"MiqProvisionRequest", :count⇒6, :ids⇒[115, 116, 104, 102, 105, 112], :cpu⇒6, :memory⇒1536, , :storage⇒62914560}
requests_by_owner, requests_by_group	Collect stats based on provision requests made the same day as the request by the same owner or LDAP group as the owner in the current request, regardless of when the provision request is scheduled to run. (Results do not include Provision Requests that have been denied.) Sample return object: {:class_name⇒"MiqProvisionRequest", :count⇒6, :ids⇒[115, 116, 104, 102, 105, 112], :cpu⇒6, :memory⇒1536, , :storage⇒62914560}
active_provisions_by_owner, active_provisions_by_group, active_provisions	Collect stats based on currently active provision requests by the same owner or LDAP group as the owner in the current request. Sample return object: {:class_name⇒"MiqProvisionRequest", :count⇒6, :ids⇒[115, 116, 104, 102, 105, 112], :cpu⇒6, :memory⇒1536, , :storage⇒62914560}

1. Supported for Red Hat Virtualization and VMware virtual machines
2. Supported for VMware virtual machines only

Terminology

- Account Role A designation assigned to a user allowing or restricting a user to parts and functions of the Infrastructure Automation console.
- Action An execution that is performed after a condition is evaluated.
- Alert Infrastructure Automation alerts notify administrators and monitoring systems of critical configuration changes and threshold limits in the virtual environment. The notification can take the form of either an email or an SNMP trap.
- Analysis Profile A customized scan of hosts, virtual machines, or instances. You can collect information from categories, files, event logs, and registry entries.
- Cloud A pool of on-demand and highly available computing resources. The usage of these resources are scaled depending on the user requirements and metered for cost.
- Infrastructure Automation appliance A virtual machine on which the virtual management database (VMDB) and Infrastructure Automation server reside.
- Infrastructure Automation Console A web-based interface into the Infrastructure Automation appliance.
- Infrastructure Automation Role A designation assigned to a Infrastructure Automation server that defines what a Infrastructure Automation server can do.
- Infrastructure Automation Server The application that runs on the Infrastructure Automation appliance and communicates with the SmartProxy and the VMDB.

- Cluster Hosts that are grouped together to provide high availability and load balancing.
- Condition A test of criteria triggered by an event.
- Discovery Process run by the Infrastructure Automation server which finds virtual machine and cloud providers.
- Drift The comparison of a virtual machine, instance, host, cluster to itself at different points in time.
- Event A trigger to check a condition.
- Event Monitor Software on the Infrastructure Automation appliance which monitors external providers for events and sends them to the Infrastructure Automation server.
- Host A computer on which virtual machine monitor software is loaded.
- Instance/Cloud Instance A on-demand virtual machine based upon a predefined image and uses a scalable set of hardware resources such as CPU, memory, networking interfaces.
- Managed/Registered VM A virtual machine that is connected to a host and exists in the VMDB. Also, a template that is connected to a provider and exists in the VMDB. Note that templates cannot be connected to a host.
- Managed/Unregistered VM A virtual machine or template that resides on a repository or is no longer connected to a provider or host and exists in the VMDB. A virtual machine that was previously considered registered may become unregistered if the virtual machine was removed from provider inventory.
- Provider A computer on which software is loaded which manages multiple virtual machines that reside on multiple hosts.
- Policy A combination of an event, a condition, and an action used to manage a virtual machine.
- Policy Profile A set of policies.
- Refresh A process run by the Infrastructure Automation server which checks for relationships of the provider or host to other resources, such as storage locations, repositories, virtual machines, or instances. It also checks the power states of those resources.
- Regions Regions are used to create a central database for reporting and charting. Regions are used primarily to consolidate multiple VMDBs into one master VMDB for reporting.
- Resource A host, provider, instance, virtual machine, repository, or datastore.
- Resource Pool A group of virtual machines across which CPU and memory resources are allocated.
- Repository A place on a datastore resource which contains virtual machines.
- SmartProxy
    - The SmartProxy is a software agent that acts on behalf of the Infrastructure Automation appliance to perform actions on hosts, providers, storage and virtual machines.
    - The SmartProxy can be configured to reside on the Infrastructure Automation appliance or on an ESX server version.
    - The SmartProxy can be deployed from the Infrastructure Automation appliance, and provides visibility to the VMFS storage. Each storage location must have a SmartProxy with visibility to it. The SmartProxy acts on behalf of the Infrastructure Automation appliance.
    - If the SmartProxy is not embedded in the Infrastructure Automation server, it communicates with the Infrastructure Automation appliance over HTTPS on standard port 443.
- SmartState Analysis Process run by the SmartProxy which collects the details of a virtual machine or instance. Such details include accounts, drivers, network information, hardware, and security patches. This process is also run by the Infrastructure Automation server on hosts and clusters. The data is stored in the VMDB.
- SmartTags Descriptors that allow you to create a customized, searchable index for the resources in your clouds and infrastructure.
- Storage Location A device, such as a VMware datastore, where digital information resides that is connected to a resource.
- Tags Descriptive terms defined by a Infrastructure Automation user or the system used to categorize a resource.
- Template A template is a copy of a preconfigured virtual machine, designed to capture installed software and software configurations, as well as the hardware configuration, of the original virtual machine.
- Unmanaged Virtual Machine Files discovered on a datastore that do not have a virtual machine associated with them in the VMDB. These files may be registered to a provider that the Infrastructure Automation server does not have configuration information on. Possible causes may be that the provider has not been discovered or that the provider has been discovered, but no security credentials have been provided.

- Virtual Machine A software implementation of a system that functions similar to a physical machine. Virtual machines utilize the hardware infrastructure of a physical host, or a set of physical hosts, to provide a scalable and on-demand method of system provisioning.
- Virtual Management Database (VMDB) Database used by the Infrastructure Automation appliance to store information about your resources, users, and anything else required to manage your virtual enterprise.
- Virtual Thumbnail An icon divided into smaller areas that summarize the properties of a resource.
- Zones Infrastructure Automation Infrastructure can be organized into zones to configure failover and to isolate traffic. Zones can be created based on your environment. Zones can be based on geographic location, network location, or function. When first started, new servers are put into the default zone.

Managing Authentication for Infrastructure management stand-alone deployment

You can find details about how to manage authentication of Infrastructure management in stand-alone mode.

Important: These instructions do not apply to Infrastructure management when installed with IBM Cloud Pak for AIOps. These instructions are only to be used when you are installing and running Infrastructure management as a stand-alone instance.

- Active Directory
- 2-Factor-Authentication with IPA
- IPA/AD Trust Authentication
- LDAP
- SAML
- OpenID-Connect

- Active Directory
- Enabling Network Manager
- Discovering AD Domains
- Joining AD Domain
- Allowing AD Users to Login
- Configure SSSD
- Configure Apache
- Configure Administrative UI

Active Directory

In this guide we will cover how to manually configure a 7.1 Appliance's external authentication to work against Active Directory. This provides AD users access to the Appliance UI as well as the REST API.

1. Enabling Network Manager
2. Discovering AD Domain
3. Joining AD Domain
4. Allowing AD Users to Login to appliance
5. Configure SSSD
6. Configure Apache
7. Configure Administrative UI

In these examples, the AD Domain shown will be EXAMPLE.COM

Enabling Network Manager

By default, the appliance is configured with Network Manager enabled. Having Network Manager enabled allows the **realm** command to discover the available Active Directory domains that can be joined. If not enabled, one can still join an AD domain if known by the domain name.

If the appliance has Network Manager disabled, it can be enabled as follows:

```
# systemctl enable NetworkManager
# systemctl start NetworkManager
```

Discovering AD Domains

This only works if Network Manager is enabled.

```
# realm discover
example.com
  type: kerberos
  realm-name: EXAMPLE.COM
  domain-name: example.com
  configured: kerberos-member
  server-software: active-directory
  client-software: sssd
  required-package: oddjob
  required-package: oddjob-mkhomedir
  required-package: sssd
  required-package: adcli
  required-package: samba-common
  login-formats: %U@example.com
  login-policy: allow-realm-logins
```

Joining AD Domain

When joining an AD domain, specify a user that has enough permission to be able to browse the directory.

```
# realm join example.com -U user
Password for user: xxxxxxxx
#
```

Allowing AD Users to Login

```
# realm permit --all
```

Configure SSSD

Update the **/etc/sssd/sssd.conf** file as follows:

Note: Starting with SSSD version 1.15.2, which will be available in CentOS version 7.4, SSSD will provide the domain name as a user attribute. The below examples show how to set *ldap_user_extra_attrs* and *user_attributes* to take advantage of this new feature. If running an appliance built with CentOS version prior to CentOS 7.4 do not include *domainname* for these attributes.

```
    [domain/example.com]
    ad_domain = example.com
    krb5_realm = EXAMPLE.COM
    realmd_tags = manages-system joined-with-samba
    cache_credentials = True
    id_provider = ad
    krb5_store_password_if_offline = True
    default_shell = /bin/bash
    ldap_id_mapping = True
    use_fully_qualified_names = True
    fallback_homedir = /home/%d/%u
    access_provider = ad
=>  ad_gpo_map_permit = +httpd-auth
=>  ldap_user_extra_attrs = mail, givenname, sn, displayname, domainname

=>  [sssd]
=>  domains = example.com
=>  config_file_version = 2
=>  services = nss, pam, ifp
=>  default_domain_suffix = example.com

=>  [nss]
=>  homedir_substring = /home

=>  [pam]
=>  default_domain_suffix = example.com

=>  [ifp]
=>  default_domain_suffix = example.com
=>  allowed_uids = apache, root, manageiq
=>  user_attributes = +mail, +givenname, +sn, +displayname, +domainname
```

Configure Apache

Make sure the Kerberos keytab created by **realm join** above is readable by Apache.

```
# chgrp apache /etc/krb5.keytab
# chmod 640    /etc/krb5.keytab
```

Create the Apache configuration files

```
# TEMPLATE_DIR="/opt/IBM/infrastructure-management-appliance/TEMPLATE"
# cp ${TEMPLATE_DIR}/etc/pam.d/httpd-auth                          \
                    /etc/pam.d/httpd-auth
# cp ${TEMPLATE_DIR}/etc/httpd/conf.d/manageiq-remote-user.conf       \
                    /etc/httpd/conf.d/
# cp ${TEMPLATE_DIR}/etc/httpd/conf.d/manageiq-external-auth.conf.erb \
                    /etc/httpd/conf.d/manageiq-external-auth.conf
```

Update the Apache configuration file **/etc/httpd/conf.d/manageiq-external-auth.conf** as follows to reference the Kerberos keytab appropriately.

```
...
    <Location /dashboard/kerberos_authenticate>
      AuthType            GSSAPI
      AuthName            "GSSAPI Single Sign On Login"
=>    GssapiCredStore     keytab:/etc/krb5.keytab
      GssapiLocalName     on
      Require             pam-account httpd-auth

      ErrorDocument 401 /proxy_pages/invalid_sso_credentials.js
    </Location>

...
```

Set appropriate SELinux permissions:

```
# setsebool -P allow_httpd_mod_auth_pam on
# setsebool -P httpd_dbus_sssd          on
```

Restart Services

```
# systemctl restart sssd
# systemctl restart httpd
```

Configure Administrative UI

Login as admin, then in *Configure→Configuration→Authentication*

- Set mode to External (httpd)
- Check: *Get User Groups from External Authentication (httpd)*
- Check: *Enable Single Signon* if you want to allow Kerberos SSO to AD.
- Click Save.

The above steps need to be done on each UI and WebService enabled appliance.

in *Configure→Configuration→Access Control*

- Make sure the user's AD group for the appliance are created and appropriate roles assigned to those groups.

- 2-Factor Authentication
- Requirements
- Configure Administrative UI
- Configure External Authentication
- Enabling 2-Factor Authentication
- Using 2-Factor Authentication

2-Factor Authentication

Enabling 2-Factor Authentication on the Appliance is currently supported with External Authentication to IPA.

In this guide we will cover how to manually configure a 7.1 Appliance's external authentication to work with 2-Factor Authentication with IPA. This provides IPA Users access to the Appliance Administrative UI and the REST API using their IPA Password followed by a One-Time-Password.

1. Requirements
2. Configuring the Administrative UI

3. Configuring External Authentication
4. Enabling 2-Factor Authentication
5. Using 2-Factor Authentication

Requirements

The following is needed in order to enable 2-Factor-Authentication to the Appliance:

- A CentOS/RHEL 7.1 based Appliance
- IPA Server based on FreeIPA 4.1.0 or later

Configure Administrative UI

Login as admin, then in *Configure→Configuration→Authentication*

- Set mode to External (httpd)
- Check: *Get User Groups from External Authentication (httpd)*
- Do Not Check: *Enable Single Signon*
  - **Note**: Kerberos SSO with OTP is not supported in the current release of FreeIPA 4.1.0 on CentOS/RHEL 7.1
- Click Save.

The above steps need to be done on each UI and WebService enabled appliance.

in *Configure→Configuration→Access Control*

- Make sure the user's groups are created on the Appliance and appropriate roles assigned to those groups.

Configure External Authentication

Using the Appliance Console, simply enable external authentication as you currently would to IPA.

1. Login to console as *root*
2. Run *appliance_console*
3. Summary screen should show **External Auth** as *not configured*, Press any key
4. From the Advanced Setting menu, select the menu item **Configure External Authentication (httpd)**
5. Enter the FQDN of the IPA Server, i.e. *ipaserver.test.company.com*
6. Enter the IPA Server domain, i.e. *test.company.com*
7. Enter the IPA Server realm, i.e. *TEST.COMPANY.COM*
8. Press enter to select the default IPA Server Principal, i.e. *admin*
9. Enter the Password of the IPA Server Principal
10. Review details, and Enter **y** to proceed.

Enabling 2-Factor Authentication

Enabling 2-Factor Authentication is done using the IPA administrative UI.

- The IPA administrator needs to change the User authentication types for the user from Password or Radius to *Two factor authentication (password + OTP)*
- The previous can be done either per user or for all users from the IPA Server global Configuration under User Options.
- The user logs into IPA, then from the OTP Tokens tab, add an OTP token - Time-based (TOTP) or Counter-based (HOTP).
- Specifying a Description for the OTP, then clicking add, brings up a QR Code which can then be scanned into an app like **FreeOTP** for example.

- Then from the main IPA login screen, the user can Synchronize the OTP Token by specifying their Username, password and then two OTP's generated from the **FreeOTP** app.
- After which, the user can login to FreeIPA using their Password+OTP.

Using 2-Factor Authentication

Once the above is done, the user now needs to specify their IPA Username and their Password followed by their One-Time-Password as generated by the **FreeOTP** app, when:

- Logging into the Appliance Administrative UI
- Accessing the REST API

- IPA/AD Trust Authentication
- Requirements
- Set up and configure an IPA server for AD Trust Authentication
- Enable the MiQ Appliance to use the configued IPA server
- Configure the MiQ appliance to use external authentication
- Create groups on the MiQ appliance
- Use AD Trust Authentication

IPA/AD Trust Authentication

Active Directory (AD) Trust Authentication on the Appliance is supported with External Authentication to IPA.

In this guide we will cover how to manually configure an Appliance's external authentication to work with AD Trust Authentication using IPA. This provides IPA Users access to the Appliance Administrative UI and the REST API using their AD credentials.

1. Requirements
2. Set up and configure an IPA server for AD Trust Authentication
3. Enable the MiQ Appliance to use the configued IPA server
4. Configure the MiQ appliance to use external authentication
5. Create groups on the MiQ appliance
6. Use AD Trust Authentication

Requirements

The following is needed in order to enable AD Trust Authentication to the Appliance:

- A CentOS/RHEL 7.2 based MiQ Appliance
- Windows Server 2008 R2 or later with configured AD DC and DNS installed locally on the DC

Set up and configure an IPA server for AD Trust Authentication

- Configure an IPA Server based on FreeIPA 3.3.3 or later

Instructions for setting up and configuring cross-realm trust between an IPA domain and an AD (Active Directory) domain can be found at freeipa.org Active Directory Trust Setup

- Add necessary user attributes to the SSSD configuration on the IPA server

The SSSD configuration file on the IPA Server must be updated to list needed user attributes.

Add the following entry to the SSSD configuration file /etc/sssd/sssd.conf

Note: Starting with SSSD version 1.15.2, which will be available in CentOS version 7.4, SSSD will provide the domain name as a user attribute. The below examples show how to set *ldap_user_extra_attrs* and *user_attributes* to take advantage of this new feature. If running an appliance built with CentOS version prior to CentOS 7.4 do not include *domainname* for these attributes.

```
[ifp]
user_attributes = +mail, +givenname, +sn, +displayname, +domainname
```

and update *ldap_user_extra_attrs* to include *domainname* where appropriate.

```
[domain/example.com]
ldap_user_extra_attrs = mail, givenname, sn, displayname, domainname
```

- DNS Configuration Significance

Special care should be made when configuring DNS as improper DNS configurations can result in poor performance and improper functionality. For more details refer to the following documents:

- freeipa.org Active Directory Trust Setup/DNS configuration
- freeipa.org DNS
- freeipa.org DNSSEC

Enable the MiQ Appliance to use the configued IPA server

Use the Appliance Console to enable external authentication to the IPA Server.

1. Log in to console as *root*
2. Run *appliance_console*
3. Summary screen should show **External Auth** as *not configured*, Press any key
4. From the Advanced Setting menu, select the menu item **Configure External Authentication (httpd)**
5. Enter the FQDN of the IPA Server, i.e. *ipaserver.test.company.com*
6. Enter the IPA Server domain, i.e. *test.company.com*
7. Enter the IPA Server realm, i.e. *TEST.COMPANY.COM*
8. Press enter to select the default IPA Server Principal, i.e. *admin*
9. Enter the Password of the IPA Server Principal
10. Review details, and Enter **y** to proceed.

Configure the MiQ appliance to use external authentication

Log in to the MiQ appliance as admin, then in *Settings→Configuration→Server→Authentication*

- Set mode to External (httpd)
- Check: *Get User Groups from External Authentication (httpd)*
- *Optionally* Check: *Enable Single Signon*
- Click Save.

The above steps need to be done on each UI and WebService enabled appliance.

in *Settings→Configuration→Access Control*

- Make sure the user's groups are created on the Appliance and appropriate roles assigned to those groups.

Create groups on the MiQ appliance

The below steps need to be done on each UI and WebService enabled appliance.

Log in to the MiQ appliance as admin, then in *Settings→Configuration→Access Control→Groups→Configuration→Add a new Group*

- Check: *(Look Up LDAP Groups)*
- Enter the AD user as the *User to Look Up* i.e. *ipauser@ipaserver.test.company.com*
- Click Retrieve.
- Choose a group from the *LDAP Groups for User* dropdown.
- Assign the appropriate roles to the group
- Click Add.

Use AD Trust Authentication

Once the above is done, the user simply needs to specify their AD Username and Password when:

- Logging into the Appliance Administrative UI
- Accessing the REST API
- Using the Self Service UI
- Using the Single Sign On (SSO) to access the MiQ appliance after generating a Kerberos ticket by using kinit with AD credentials.

- LDAP
- Testing OpenLDAP
  - With SSL
  - With User Bind DN and Password
- Configure SSSD
  - Configure sssd.conf with authselect (CentOS 8 and newer)
  - Configure sssd.conf with authconfig (CentOS 7 and older)
  - Update sssd.conf
    - Testing SSSD Updates
- Configure Apache
- Configure SELinux
- Configure Administrative UI

LDAP

This guide covers how to manually configure an appliance's external authentication to work against LDAP. This provides LDAP users access to the Appliance UI as well as the REST API.

1. Testing OpenLDAP
2. Configure SSSD
3. Configure Apache
4. Configure SELinux
5. Configure Administrative UI

In these examples, the LDAP domain used will be *example.com*

Note: The SSSD and OpenLDAP configurations shown below are simply examples. Depending on the Ldap environment, Ldap directory server used, the configurations can widely differ. Many other options are available to the *authconfig* command as well as the *sssd.conf* configuration file.

The following reference are available for customizing SSSD for any LDAP directory:

```
# man authconfig
# man ldap.conf
# man sssd-ldap
```

The document @ https://onemoretech.wordpress.com/2014/02/23/sssd-for-ldap-auth-on-linux/ is also helpful.

Testing OpenLDAP

With SSL

For Configuring external authentication to Ldap, the sssd-ldap communication must be secured (SSL or Start TLS). So an SSL Certificate needs to be created.

The *apacheds-cert.pem* certificate file is used in the examples below.

Update the LDAP Directory Server to point to the keystore for the certificate. This is used for both SSL and Start TLS.

For SSL, update the **/etc/openldap/ldap.conf** accordingly to test:

```
SASL_NOCANON    on

URI             ldaps://ldap-example.com:10636
BASE            dc=example,dc=com
TLS_REQCERT     demand
```

```
TLS_CACERTDIR   /etc/openldap/cacerts
TLS_CACERT      /etc/openldap/cacerts/apacheds-cert.pem
```

Then try out a search:

```
# ldapsearch -x -H ldaps://ldap-example:10636 -LLL \
       -b "ou=people,dc=example,dc=com" -s sub     \
       "(objectclass=organizationalPerson)"
```

With User Bind DN and Password

You can also test the ldap directory by specifying a user Bind DN and password on the Ldap search command line as follows:

```
# ldapsearch -x -H ldap://ldap-example:389 -LLL                 \
       -b "ou=people,dc=example,dc=com" -s sub                  \
       -D "ui=:userid,ou=People,dc=example,dc=com" -w :password \
       "(objectclass=organizationalPerson)"
```

Configure SSSD

There are two different ways that SSSD can be configured.

For *CentOS 7* and earlier **authconfig** is used to configure SSSD. Starting with *CentOS 8* a new command, **authselect** is used.

Perform the steps in only one of these two sections below:

- Configure sssd.conf with authselect (CentOS 8 and newer)
- Configure sssd.conf with authconfig (CentOS 7 and older)

Note: SSSD must be configured to communicate with LDAP over SSL or Start TLS. Otherwise an authentication error will be returned to the Appliance login screen.

Configure sssd.conf with authselect (CentOS 8 and newer)

```
  # authselect select sssd --force
```

Seed the sssd.conf configuration file

authselect does not automatically generate the */etc/sssd/sssd.conf* file, however the template, *sssd.conf.erb*, is now delivered with ManageIQ which can be used to seed the */etc/sssd/sssd.conf*.

```
# TEMPLATE_DIR="/opt/IBM/infrastructure-management-appliance/TEMPLATE"
# mkdir -p /etc/sssd
# cp ${TEMPLATE_DIR}/etc/sssd/sssd.conf.erb /etc/sssd/sssd.conf
# chmod 600 /etc/sssd/sssd.conf
```

Update the SSSD configuration file /etc/sssd/sssd.conf to specify the correct *ldap_search_base* and *ldap_uri*:

```
...
ldap_search_base = dc=example,dc=com
id_provider = ldap
auth_provider = ldap
chpass_provider = ldap
ldap_uri = ldaps://ldap-example.com:10636
ldap_id_use_start_tls = False
cache_credentials = True
...
```

Configure sssd.conf with authconfig (CentOS 7 and older)

```
  # authconfig \
    --enablesssd                                \
    --enablesssdauth                            \
    --enablelocauthorize                        \
    --enableldap                                \
    --enableldapauth                            \
    --ldapserver=ldaps://ldap-example.com:10636 \
    --disableldaptls                            \
    --ldapbasedn=dc=example,dc=com              \
    --enablerfc2307bis                          \
    --enablecachecreds                          \
    --update
```

Update sssd.conf

Edit the different sections in **/etc/sssd/sssd.conf** for the Appliance as in the following example, customizing the main **[domain/example.com]** section for the particular Ldap installation.

Note: Starting with SSSD version 1.15.2, which will be available in CentOS version 7.4, SSSD will provide the domain name as a user attribute. The below examples show how to set *ldap_user_extra_attrs* and *user_attributes* to take advantage of this new feature. If running an appliance built with CentOS version prior to CentOS 7.4 do not include *domainname* for these attributes.

```
=>  [domain/example.com]
    autofs_provider = ldap
    id_provider = ldap
    auth_provider = ldap
    chpass_provider = ldap
    ldap_schema = rfc2307bis

    ldap_uri = ldaps://ldap-example.com:10636
    ldap_id_use_start_tls = False
=>  ldap_tls_cacertdir = /etc/openldap/cacerts
=>  ldap_tls_cacert = /etc/openldap/cacerts/apacheds-cert.pem

=>  ldap_pwd_policy = none

    ldap_search_base = dc=example,dc=com
=>  ldap_network_timeout = 3

=>  ldap_user_search_base = ou=people,dc=example,dc=com
=>  ldap_user_object_class = posixAccount
=>  ldap_user_name = uid
=>  ldap_user_uid_number = uidNumber

=>  ldap_group_object_class = groupOfNames
=>  ldap_group_search_base = ou=user_groups,dc=example,dc=com
=>  ldap_group_name = cn
=>  ldap_group_member = member

    cache_credentials = True
=>  entry_cache_timeout = 600

=>  ldap_user_extra_attrs = mail, givenname, sn, displayname, domainname

    [sssd]
=>  domains = example.com
    config_file_version = 2
=>  services = nss, pam, ifp
=>  sbus_timeout = 30
=>  default_domain_suffix = example.com

    [nss]
    homedir_substring = /home

    [pam]
=>  default_domain_suffix = example.com

=>  [ifp]
=>  default_domain_suffix = example.com
=>  allowed_uids = apache, root, manageiq
=>  user_attributes = +mail, +givenname, +sn, +displayname, +domainname
```

Testing SSSD Updates

The best way to test manual SSSD updates is to query a user via the message bus system (D-Bus) API. When user attributes and groups are properly returned, that is a good indication that the External Authentication would succeed.

For the *dbus-send* command to succeed when run from command line, SELinux must be in permissive mode.

```
# setenforce 0
```

After each update to **sssd.cond**, restart the sssd service before testing the changes.

```
systemctl restart sssd
```

Example query of user attributes for user evmuser. This primarily validates the *ldap_user_* attributes of **sssd.conf**.

```
# dbus-send --print-reply --system --dest=org.freedesktop.sssd.infopipe /org/freedesktop/sssd/infopipe
org.freedesktop.sssd.infopipe.GetUserAttr string:evmuser
array:string:mail,givenname,sn,displayname,domainname
```

Query groups of user evmuser. This primarily validates the *ldap_group_* attributes of **sssd.conf**.

```
# dbus-send --print-reply --system --dest=org.freedesktop.sssd.infopipe /org/freedesktop/sssd/infopipe
org.freedesktop.sssd.infopipe.GetUserGroups string:evmuser
```

With logging for each of the **sssd.conf** section in the appropriate /var/log/sssd/ log file. The log files of primary interest are:

Log File	Type of debugging
/var/log/sssd/sssd.log	SSSD communication with processes
/var/log/sssd/sssd_example.com.log	sssd-ldap communication to the LDAP server
/var/log/sssd/sssd_ifp.log	Gathering user and group information from LDAP server

For each of the log file here, an debug level attribute (maximum value of 9) can be added to the appropriate **sssd.conf** section as follows:

```
    [ifp]
=>  debug_level = 9
    default_domain_suffix = example.com
    ...
```

When done testing, re-enable SELinux enforcing mode.

```
# setenforce 1
```

Note: When testing authentication on the Appliance, if updates are done in LDAP and not immediately seen when authenticating, then clean the SSSD cache as follows then retry:

```
# sss_cache -E
```

Configure Apache

Create the Apache configuration files

```
# TEMPLATE_DIR="/opt/IBM/infrastructure-management-appliance/TEMPLATE"
# cp ${TEMPLATE_DIR}/etc/pam.d/httpd-auth                          \
                    /etc/pam.d/httpd-auth
# cp ${TEMPLATE_DIR}/etc/httpd/conf.d/manageiq-remote-user.conf      \
                    /etc/httpd/conf.d/
# cp ${TEMPLATE_DIR}/etc/httpd/conf.d/manageiq-external-auth.conf.erb \
                    /etc/httpd/conf.d/manageiq-external-auth.conf
```

Configure SELinux

For SSSD to Ldap, assure that non-standard Ldap ports (other than 389 or 636) are allowed:

```
# semanage port -a -t ldap_port_t -p tcp 10389
# semanage port -a -t ldap_port_t -p tcp 10636
```

Set appropriate SELinux permissions:

```
# setsebool -P allow_httpd_mod_auth_pam on
# setsebool -P httpd_dbus_sssd          on
```

Restart Services

```
# systemctl restart sssd
# systemctl restart httpd
```

Configure Administrative UI

Log in as admin, navigate to *Settings* > *Application Settings* > *Settings*. Select the server, then select the *Authentication tab*

- Set mode to External (httpd)
- Check: *Get User Groups from External Authentication (httpd)*
- Do not check: *Enable Single Signon* since Kerberos is not configured against LDAP.
- Click Save.

The above steps need to be completed on each UI and WebService enabled appliance.

Navigate to *Settings* > *Application Settings* > *Access Control*

- Make sure the user's LDAP group for the appliance are created and appropriate roles assigned to those groups.

- SAML
- Requirements
- Configure SAML
    - Apache Configuration
    - SAML Configuration
        - SAML Assertions

- Configure Administrative UI
- Re-Enabling Local Login
  - Administrative UI:
  - Appliance Console Interface:
  - Appliance Console CLI:
- Configuring SAML using the CLI
  - Configure SAML
  - Unconfigure SAML

SAML

External Authentication on the Appliance has been enhanced to support SAML. The SAML implementation has been tested with KeyCloak 1.8.

In this guide we will cover how to manually configure an Appliance's external authentication to work with SAML. The SAML implementation has been tested with KeyCloak but is implemented generically using Apache's mod_auth_mellon module and should work with other SAML Identity Providers.

The current implementation only secures the Appliance's Web administrative UI with SAML.

Note: The REST API and Self-Service UI do not currently support SAML.

1. Requirements
2. Configuring SAML
3. Configuring the Administrative UI
4. Re-Enabling Local Login
5. Configuring SAML using the CLI

Requirements

The following is needed in order to enable SAML Authentication to the Appliance:

- A CentOS/RHEL 7.2 based Appliance
- A SAML Identity Provider, (e.g. KeyCloak 1.8 or Later)

Configure SAML

All SAML related certificates and keys are accessed from /etc/httpd/saml2/

First ssh to the appliance as root, then create that directory:

```
# mkdir -p /etc/httpd/saml2
```

Apache Configuration

Copy the Apache remote user and SAML template configuration files:

```
# TEMPLATE_DIR="/opt/IBM/infrastructure-management-appliance/TEMPLATE"
# cp ${TEMPLATE_DIR}/etc/httpd/conf.d/manageiq-remote-user.conf        \
    /etc/httpd/conf.d/
# cp ${TEMPLATE_DIR}/etc/httpd/conf.d/manageiq-external-auth-saml.conf \
    /etc/httpd/conf.d/
```

SAML Configuration

Notable defaults in the *manageiq-external-auth-saml.conf* file:

Description	Default
Identity Provider Files (i.e. KeyCloak)	
Metadata File	/etc/httpd/saml2/idp-metadata.xml
Service Provider Files (i.e. mod_auth_mellon)	
Private Key File	/etc/httpd/saml2/miqsp-key.key
Certificate File	/etc/httpd/saml2/miqsp-cert.cert
Metadata File	/etc/httpd/saml2/miqsp-metadata.xml

Other mod_auth_mellon parameters must not be modified as the Appliance expects them to be defined as such, i.e. End points, Protected URL, etc.

For the Service Provider files on the Appliance, these can be generated using the mod_auth_mellon command *mellon_create_metadata.sh* as follows:

```
# cd /etc/httpd/saml2
# /usr/libexec/mod_auth_mellon/mellon_create_metadata.sh https://<miq-appliance> https://<miq-appliance>/saml2
```

The mellon_create_metadata.sh script creates file names based on the appliance URL but can be renamed to match the expected file names from the *manageiq-external-auth-saml.conf* file:

```
# mv https_<miq-appliance>.key  miqsp-key.key
# mv https_<miq-appliance>.cert miqsp-cert.cert
# mv https_<miq-appliance>.xml  miqsp-metadata.xml
```

With the service provider metadata.xml file generated, the Service Provider definition can now be defined in the Identity Provider.

For KeyCloak, a Realm can be created for one or more Appliances with individual Clients defined one per Appliance where the Client ID is essentially the URL of the appliance.

Adding a Client in the KeyCloak ManageIQ Realm:

- Select and Import the miqsp-metadata.xml file created for mod_auth_mellon.
- Set Client ID as https://<miq-appliance>
- Set Client protocol as **saml**.

The Client definition for the appliance can then be updated with the following:

Setting	Value
Name ID Format	username
Valid Redirect URIs	https://<miq-appliance/saml2/paosResponse>
	https://<miq-appliance/saml2/postResponse>
Base URL	https://<miq-appliance/>

Fine Grain SAML Endpoint Configuration	
Setting	**Value**
Assertion Consumer Service POST Binding URL	https://<miq-appliance/saml2/postResponse>
Assertion Consumer Service Redirect Binding URL	https://<miq-appliance/saml2/postResponse>
Logout Service Redirect Binding URL	https://<miq-appliance/saml2/logout>

The Identity Provider's Metadata file idp-metadata.xml can then be obtained as follows:

```
# cd /etc/httpd/saml2
# curl -s -o idp-metadata.xml \
    http://<keycloak-server>:8080/auth/realms/<miq-realm>/protocol/saml/descriptor
```

Finally, restart Apache on the appliance as follows:

```
# systemctl restart httpd
```

SAML Assertions

For authenticating to the appliance, the following remote user parameters are looked at by the appliance upon a successful login and redirect from the Identity Provider.

HTTP Environment	SAML Assertion
REMOTE_USER	username
REMOTE_USER_EMAIL	email
REMOTE_USER_FIRSTNAME	firstname
REMOTE_USER_LASTNAME	lastname
REMOTE_USER_FULLNAME	fullname
REMOTE_USER_GROUPS	groups

For KeyCloak, the above SAML Assertions can be created for the Appliance Client in KeyCloak as Mappers.

Name	Mapper Type	Property	Friendly Name	SAML Attribute Name	SAML Attribute Name Format
username	User Property	username	username	username	Basic
email	User Property	email	email	email	Basic
firstname	User Property	firstName	firstname	firstname	Basic
lastname	User Property	lastName	lastname	lastname	Basic
Name	**Mapper Type**	**User Attribute**	**Friendly Name**	**SAML Attribute Name**	**SAML Attribute Name Format**

Name	Mapper Type	User Attribute	Friendly Name	SAML Attribute Name	SAML Attribute Name Format
fullname	User Attribute	fullName	fullname	fullname	Basic

Name	Mapper Type	Group attribute name	Friendly Name	SAML Attribute Name	SAML Attribute Name Format
groups	Group List	groups	groups	groups	Basic

Note: The fullName attribute was not available in the default database as of this writing and must be added to each user as a user attribute.

Configure Administrative UI

After having configured Apache for SAML, the next step is to update the Appliance Administrative UI to be SAML aware and function accordingly.

Login as admin, then in *Configure→Configuration→Authentication*

- Set mode to External (httpd)
- Check: *Enable Single Signon* - With this option enabled, initial access to the Appliance Administrative UI will redirect to the SAML Identity Provider authentication screen. Note that logouts from the Appliance will return the user to the Appliance login screen allowing them to login as admin unless *Disable Local Login* is checked below.
- Check: *provider Type: _Enable SAML* - This enables the SAML login button on the login screen, the redirects to the SAML protected page for authentication, and supports the SAML logout process.
- Optional: Check: *Disable Local Login* - Do this **only** if you need to disable *admin* login to appliance and only allow SAML based authentication. Note that if there are issues with the Identity Provider or you need admin access to the appliance you won't be able to login until you re-enable the Local Login as described below.
- Check: *Get User Groups from External Authentication (httpd)*
- Click Save.

The above steps need to be done on each UI enabled appliance.

in *Configure→Configuration→Access Control*

- Make sure the user's groups are created on the Appliance and appropriate roles assigned to those groups.

Re-Enabling Local Login

If the Local Login has been disabled in the Administrative UI and there is a need to be able to login as *admin*, the Local Login can be re-enabled as follows:

Administrative UI:

This option is available if the Identity Provider is available and one can login using a user with enough administrative privileges to update it:

- Login as administrative user,
- In then in *Configure→Configuration→Authentication* uncheck *Disable Local Login* and save.

Appliance Console Interface:

- ssh to the appliance as root
- Run *appliance_console*
- Select menu entry *Update External Authentication Options*
- Select *Enable Local Login*
- then *Apply updates*

Appliance Console CLI:

- ssh to the appliance as root
- Run *appliance_console_cli –extauth-opts* **local_login_disabled=false**

Configuring SAML using the CLI

Another way to configure SAML on the appliance is to use the appliance console CLI instead of the above steps. The **`--saml-config`** subcommand of the appliance console CLI allows one to configure the appliance for SAML Authentication by updating both Apache configurations

as well as the necessary Administrative UI settings. The **--saml-unconfig** subcommand is also provided for unconfiguring the appliance SAML Authentication and reverting the appliance to Database Authentication.

Usage of the new Appliance Console CLI subcommands are as follows:

Configure SAML

To configure the appliance for SAML Authentication:

```
# appliance_console_cli --saml-config [--saml-client-host=miq_appliance.fqdn] \
                                       --saml-idp-metadata=file|url           \
                                      [--saml-enable-sso]
```

When –saml-client-host is not specified, the configured appliance host is used for creating the SP metadata. This host fqdn must be reachable from the SAML IDP.

Configuring SAML requires the IDP metadata file. The –saml-idp-metadata option allows the user to specify the file path of a copy that was downloaded to the appliance or by a URL, in which case the CLI downloads the copy from the IDP.

Examples:

```
--saml-idp-metadata=/tmp/downloaded_idp_metadata.xml
```

```
--saml-idp-metadata=http://<idp-host:port>/auth/realms/<miq-realm>/protocol/saml/descriptor
```

In both cases, the IDP metadata file is copied to **/etc/httpd/saml2/idp-metadata.xml**

By default, SSO is not enabled, so from the Appliance login page, the user clicks on Log In to Corporate System to get redirected to the SAML login page. With this option enabled, the Appliance redirects the user to the SAML login page for logging in.

After SAML is configured, the SP metadata file can be fetched from **/etc/httpd/saml2/miqsp-metadata.xml** to create the related Client on the SAML IDP.

Unconfigure SAML

To unconfigure the appliance from SAML Authentication and revert to Database authentication:

```
# appliance_console_cli --saml-unconfig
```

This will remove the Apache external authentication SAML configuration files and revert the appliance's authentication settings to Database mode.

- OpenID-Connect
- Requirements
- Configuring Keycloak
  - OIDC Assertions
  - Apache Configuration
  - OIDC Configuration
- Configuring the Administrative UI
- Re-Enabling Local Login
  - Administrative UI:
  - Appliance Console Interface:
  - Appliance Console CLI:
- Configuring OIDC using the CLI
  - Configure OpenID-Connect
  - Unconfigure OpenID-Connect

OpenID-Connect

External Authentication on the Appliance has been enhanced to support OpenID-Connect, which will also be referred to as *OIDC*.

The OIDC implementation has been tested with KeyCloak 3.2.1.

In this guide we will cover how to manually configure an Appliance's external authentication to work with OIDC. The OIDC implementation has been tested with KeyCloak but is implemented generically using Apache's mod_auth_openidc module and should work with other OIDC Identity Providers.

The current implementation secures the Appliance's Web administrative and Self-Service UI with OIDC.

1. Requirements
2. Configuring Keycloak
3. OIDC Assertions

4. Apache Configuration
5. OIDC Configuration
6. Configuring the Administrative UI
7. Re-Enabling Local Login
8. Configuring OIDC using the CLI

Requirements

The following is needed in order to enable OIDC Authentication to the Appliance:

- A CentOS/RHEL 7.5 based Appliance
- A OIDC Identity Provider, (e.g. KeyCloak 3.2.1 or Later)

Configuring Keycloak

For KeyCloak, a Realm can be created for one or more Appliances with individual Clients defined one per Appliance where the Client ID identifies the individual appliance, e.g. the appliance hostname.

Adding a Client in the KeyCloak ManageIQ Realm:

- Set Client ID as <miq-appliance>
- Set Client protocol as **openid-connect**.

The Client definition for the appliance can then be updated with the following:

Setting	Value
Client Protocol	openid-connect
Access Type	confidential
Valid Redirect URIs	https://<miq-appliance>
	https://<miq-appliance/ui/service/>
	https://<miq-appliance/oidc_login/redirect_uri>
User Info Signed	unsigned
Request Object Signature Algorithm	any

Note: The trailing / for the /ui/service/ entry is required.

The value used for OIDCClientSecret (*oidc_client_secret*) in the *manageiq-external-auth-oidc.conf* file is taken from the Keycloak Client's *Credentials / Secret* value. To generate a new client secret, First set the *Credentials / Client Authenticator* to *Client Id and Secret* . Then *Regenerate Secret* button.

OIDC Assertions

For authenticating to the appliance, the following remote user parameters are looked at by the appliance upon a successful login and redirect from the Identity Provider.

HTTP Environment	OIDC Assertion
REMOTE_USER	username
REMOTE_USER_EMAIL	email
REMOTE_USER_FIRSTNAME	firstname
REMOTE_USER_LASTNAME	lastname
REMOTE_USER_FULLNAME	fullname
REMOTE_USER_GROUPS	groups

For KeyCloak, the above OIDC Assertions can be created for the Appliance Client in KeyCloak as Mappers.

The following mappers are automatically generated when the OIDC client is created:

Name	Category	Mapper Type
email	Token mapper	User Property
family name	Token mapper	User Property
given name	Token mapper	User Property
full name	Token mapper	User's full name
username	Token mapper	User Property

The following *Group Membership* mapper must be manually created:

Name	Consent Required	Mapper Type	Token Claim Name	Full group path	Add to ID token	Add to access token	Add to userinfo
groups	OFF	Group Membership	groups	OFF	ON	ON	OFF

The following *User Session Note* mappers must be manually created:

Name	Consent Required	Mapper Type	User Session Note	Token Claim Name	Claim JSON Type	Add to ID token	Add to access token
Client Host	OFF	User Session Note	clientHost	clientHost	String	ON	ON
Client ID	OFF	User Session Note	clientId	clientId	String	ON	ON
Client IP Address	OFF	User Session Note	clientAddress	clientAddress	String	ON	ON

Apache Configuration

Copy the Apache OIDC template configuration files:

```
# TEMPLATE_DIR="/opt/IBM/infrastructure-management-appliance/TEMPLATE"
# cp ${TEMPLATE_DIR}/etc/httpd/conf.d/manageiq-remote-user-openidc.conf \
    /etc/httpd/conf.d/
# cp ${TEMPLATE_DIR}/etc/httpd/conf.d/manageiq-external-auth-openidc.conf.erb \
    /etc/httpd/conf.d/manageiq-external-auth-openidc.conf
```

OIDC Configuration

The Apache */etc/httpd/conf.d/manageiq-external-auth-openidc.conf* configuration files must be updated with installation specific values:

The defaults in the *manageiq-external-auth-oidc.conf* file are:

Name	Value
LogLevel	warn
OIDCCryptoPassphrase	sp-cookie
OIDCOAuthRemoteUserClaim	username

Installation specific values must be specified in the *manageiq-external-auth-oidc.conf* file for the following parameters, where the *parameter* including the <%= ... %> anchors need to be replaced with the appropriate values:

Name	Parameter	Value
ServerName and OIDCRedirectURI	miq_appliance	FQDN of the appliance
OIDCProviderMetadataURL	oidc_provider_metadata_url	The OpenID-Connect Provider URL, i.e. http://<keycloak_server>:8080/auth/realms/miq/.well-known/openid-configuration
OIDCClientID and OIDCOAuthClientID	oidc_client_id	The OpenID-Connect Client ID
OIDCClientSecret end OIDCOAuthClientSecret	oidc_client_secret	The OpenID-Connect Client Secret
OIDCOAuthIntrospectionEndpoint	oidc_introspection_endpoint	The OpenID-Connect Introspection Endpoint

Optional parameters can be appended to the end fo the configuration file to avoid SSL verification. These can be used when a valid SSL server certificate is not available for communicating with the OpenID Provider.

NOTE: Using these parameters is insecure and should only be used in a development environment.

Name	Value
OIDCSSLValidateServer	Off
OIDCOAuthSSLValidateServer	Off

Finally, restart Apache on the appliance as follows:

```
# systemctl restart httpd
```

Configuring the Administrative UI

After having configured Apache for OIDC, the next step is to update the Appliance Administrative UI to be OIDC aware and function accordingly.

Login as admin, then in *Configure→Configuration→Authentication*

- Set mode to External (httpd)
- Check: *Provider Type: Enable Enable OpenID-Connect* - This enables the OIDC login button on the login screen, that redirects to the OIDC protected page for authentication, and supports the OIDC logout process.
- Optional: Check: *Enable Single Signon* - With this option enabled, initial access to the Appliance Administrative UI will redirect to the OIDC Identity Provider authentication screen. Note that logouts from the Appliance will return the user to the Appliance login screen allowing them to login as admin unless *Disable Local Login* is checked below.
- Optional: Check: *Disable Local Login* - Do this **only** if you need to disable *admin* login to appliance and only allow OIDC based authentication. Note that if there are issues with the Identity Provider or you need admin access to the appliance you won't be able to login until you re-enable the Local Login as described below.
- Check: *Get User Groups from External Authentication (httpd)*
- Click Save.

The above steps need to be done on each UI enabled appliance.

In Configure→Configuration→Access Control

- Make sure the user's groups are created on the Appliance and appropriate roles assigned to those groups.

Re-Enabling Local Login

If the Local Login has been disabled in the Administrative UI and there is a need to be able to login as *admin*, the Local Login can be re-enabled using either of the following methods:

Administrative UI:

This option is available if the Identity Provider is available and one can login using a user with enough administrative privileges to update it:

- Login as administrative user,
- In then in *Configure→Configuration→Authentication* uncheck *Disable Local Login* and save.

Appliance Console Interface:

- ssh to the appliance as root
- Run *appliance_console*
- Select menu entry *Update External Authentication Options*
- Select *Enable Local Login*
- then *Apply updates*

Appliance Console CLI:

- ssh to the appliance as root
- Run *appliance_console_cli –extauth-opts* **local_login_disabled=false**

Configuring OIDC using the CLI

Another way to configure OpenID-Connect on the appliance is to use the appliance console CLI instead of the above steps. The **`--oidc-config`** subcommand of the appliance console CLI allows one to configure the appliance for OpenID-Connect Authentication by updating both Apache configurations as well as the necessary Administrative UI settings. The **`--oidc-unconfig`** subcommand is also provided for unconfiguring the appliance OpenID-Connect Authentication and reverting the appliance to Database Authentication.

Usage of the new Appliance Console CLI subcommands are as follows:

Configure OpenID-Connect

To configure the appliance for OpenID-Connect Authentication:

```
# appliance_console_cli --oidc-config
                        [--oidc-client-host]
                        --oidc-url
                        --oidc-client-id
                        --oidc-client-secret
                        [--oidc-introspection-endpoint]
                        [--oidc-enable-sso]
                        [--oidc-insecure]
```

When the **--oidc-introspection-endpoint** is not specified, an attempt will be made to fetch the introspection endpoint by querying the OpenID-Connect Provider URL.

When the **--oidc-client-host** is not specified, the configured appliance host is used for the ServerName and OpenID-Connect Redirect URI in the OpenID-Connect Apache configuration file. The **--oidc-client-host** is usually the externally reachable FQDN of the appliance.

When **--oidc-insecure** is specified no SSL verification will be done when attempting to fetch the introspection endpoint and SSL verification will be disabled using parameters **OIDCSSLValidateServer** and **OIDCOAuthSSLValidateServer** as described above.

Configuring OpenID-Connect requires the following parameters:

Parameter	Description
--oidc-url	This is the OpenID-Connect Provider URL
--oidc-client-id	The OpenID-Connect Provider Cliend ID
--oidc-client-secret	The OpenID-Connect Provider Client Secret

The **--oidc-url** varies for different providers. An example provider URL for keycloak:

```
--oidc-url=http://keycloak-server:8080/auth/realms/miq-realm/.well-known/openid-configuration
```

With **miq-realm** being the Keycloak Realm this OpenID-Connect appliance client is being configured for.

The **--oidc-client-id** is the Client ID specified for the appliance in the OpenID-Connect provider, usually just **<miq-appliance-fqdn>**

By default, SSO is not enabled, so from the Appliance login page, the user clicks on Log In to Corporate System to get redirected to the OpenID-Connect login page. With the **--oidc-enable-sso** option specified, the Appliance redirects the user to the OpenID-Connect login page for logging in.

Unconfigure OpenID-Connect

To unconfigure the appliance from OpenID-Connect Authentication and revert to Database authentication:

```
# appliance_console_cli --oidc-unconfig
```

This will remove the Apache external authentication OpenID-Connect configuration files and revert the appliance's authentication settings to Database mode.

- Scripting Actions in Infrastructure Automation
  - Updates to Rails and Ruby
    - Updates and Changes in Rails
    - Updates and Changes in Ruby
  - Understanding the Automate Model
    - Automate Model
    - Creating a Domain
    - Editing a Domain
    - Deleting a Domain
    - Importing a Domain from a Git Repository
    - Changing Priority Order of Domains
    - Creating a Namespace
    - Creating a Class
    - Creating a Schema for a Class
    - Editing a Field in a Schema
    - Editing Schema Sequence
    - Adding an Instance to a Class
    - Copying a Class or Instance
    - Relationships
    - Methods
      - Creating a Method
      - Creating a Dynamic Content Dialog
      - Creating a Playbook Automate Method
        - Passing variables between successive playbook methods

        - Expression Methods
            - Input Parameters
                - Optional Input Parameters
            - Creating an Expression Method
        - Simulation
            - Simulating an Automate Model
        - Importing, Exporting, and Resetting the Datastore
            - Exporting All Datastore Classes
            - Importing Datastore Classes
            - Resetting Datastores to the Default
    - Invoking Automate
        - Automate Examples
        - Invoking Automate using a Custom Button
        - Creating a Custom Button Group
        - Creating a Custom Button
        - Creating an Ansible Playbook Button
        - Editing a Custom Button
        - Deleting a Custom Button
        - Setting Enablement and Visibility for Custom Buttons
        - Using a Custom Button
        - Initiating Automate from an Event
        - Creating a Policy for Automate
        - Creating a Custom Automate Action
- Objects
    - Virtual Machine Properties
    - Methods for use in Ruby Scripts
    - Host Properties
    - Provider Properties
    - Storage Properties
- FAQs and Flows
    - Phase 1: Create Provision Request
    - Phase 2: Request Approval
    - Phase 3: Quota Validation
    - Phase 4: Provisioning
    - Phase 5: Retirement
- Inline Method to Create a Provision Request
    - Ruby method
- Migrating Custom Buttons
    - Migrating Custom Buttons

Scripting Actions in Infrastructure Automation

The Automate model provides flexibility to not only change parts of the provisioning process, but also to allow you to automate other operational tasks. Below are some scenarios where the Automate model can help accomplish these tasks.

- Intelligent Workload Management - An enterprise had a requirement that when a virtual machine has reached a High CPU Percent Ready for a specified period of time, a vMotion should occur to a more suitable host. For this reason, VMware's Distributed Resource Scheduler (DRS) was not practical, as the CPU Ready metric could not trigger DRS. The solution was to leverage Infrastructure Automation Control and Infrastructure Automation Automate to drive the management of this workflow.
- Power on only during business hours - An organization which gave a group of self-service users Infrastructure Automation access had a requirement to only allow certain virtual machines to be powered during business hours. This was solved with Infrastructure Automation Automate.
- Auto-Tagging virtual machines based on file contents - An IT organization needed a way to consume information from a text file on a virtual machine and dynamically populate vCenter. The data used to auto-tag virtual machines is also used to align unmanaged virtual machines to the business.

Updates to Rails and Ruby

Changes within Ruby and Rails versions can create issues for custom code and automation in Infrastructure Automation environments.

Before proceeding with major upgrades that include changes to versions of Ruby and Rails, review the below resources.

Updates and Changes in Rails

See Rails Releases for information on each release of Rails. To compare versions of Rails, see Comparing changes in Rails.

Updates and Changes in Ruby

See Ruby Releases for information on each release of Ruby by version number. To compare versions of Ruby, see Comparing changes in Ruby.

Understanding the Automate Model

Automate enables real-time, bi-directional process integration. This provides users with a method to implement adaptive automation for management events and administrative or operational activities.

Automate Model

The Automate model is arranged to provide an object oriented hierarchy to control automation functions. The model uses the following organizational units arranged in a hierarchy:

- **Datastore** - The main organization unit that stores the entire model.
- **Domains** - Domains act as collection of automation functions. Functions are executed depending on the order of Domain priority, which means a function in a Domain with a higher priority overrides the same functions specified in a lower-priority Domain. This allows Infrastructure Automation to specify a core Domain (ManageIQ) but allow users to override automate functions with custom Domains. Each Domain contains a set of Namespaces.
- **Namespaces** - Containers that organize and categorize functions of the model. Namespaces can contain child Namespaces as well as Classes.
- **Classes** - Templates for a specific function of the model. Each Class uses a Schema to apply to Instances to populate with default values. Each class also can contain a set of methods.
- **Instances** - An instance is a version of a class populated with initial configuration data. An instance can include a collection of any number of attributes, calls to methods, and relationships.
- **Methods** - Methods are functions within the model. Methods use Ruby code to execute various operations needed for a Class.

Infrastructure Automation contains a preconfigured Domain for users:

- **ManageIQ** - The core domain for Infrastructure Automation Automate operations. This domain is locked with the following Namespaces:
  - **Cloud** - General cloud instance lifecycle from provisioning, retirement, methods, email.
  - **Control** - Control contains email alerts for policy controls.
  - **Infrastructure** - General infrastructure VM lifecycle from provisioning, retirement, methods, email.
  - **Integration** - Used to interface with systems outside of Infrastructure Automation. Use this namespace to integrate with additional systems.
  - **Service** - Service lifecycle from provisioning, retirement, methods, email.
  - **System** - System contains classes that can provide the start points for all Infrastructure Automation Automate activities.

You can copy classes and instances from locked Domains into your own custom domains.

Note:

Changing the existing classes or instances shipped with the product is not recommended because this may hinder the operation of Infrastructure Automation. You can link to these methods using relationships.

To reset the Automate model to default settings, navigate to **Automate > Import/Export** and click the **Reset** option.

Creating a Domain

1. Navigate to **Automation > Automate > Explorer**. The default view is the Datastore.
2. Click (**Configuration**), then (**Add a New Domain**).
3. Type in a unique **Name** and **Description**. Choose if the Domain is **Enabled**.
4. Click **Add**.

The new domain is created.

Editing a Domain

1. Navigate to **Automation > Automate > Explorer**. The default view is the Datastore.
2. Select the Domain you want to edit.
3. Click (**Configuration**), then (**Edit Selected Domain**).
4. Make the required edits.
5. Click **Save**.

You have edited the selected domain.

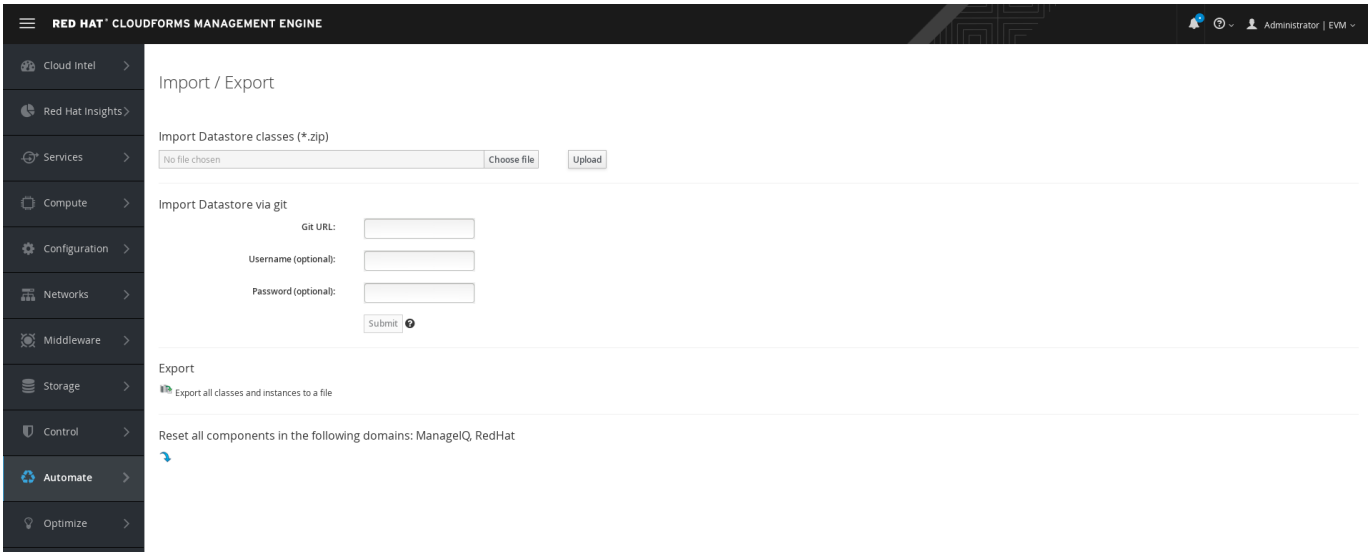
Deleting a Domain

1. Navigate to **Automation > Automate > Explorer**. The default view is the Datastore.
2. Select the **Domain** that you want to delete.
3. Click (**Configuration**), then (**Remove This Domain**).
4. A window to confirm the removal of Domain appears.
5. Click **OK**.

The selected Domain is deleted.

Importing a Domain from a Git Repository

Infrastructure Automation can import an Automate domain from a Git repository by specifying a repository and branch, along with user details. Currently, you can only add git domains via the **Import/Export** area of the user interface.



1. Navigate to **Automation > Automate > Import/Export**.
2. In **Import Datastore via git**, enter the **Git URL**. Select the branch or tag to use.
3. Optionally, enter a **Username** and **Password**.
4. Select **Verify Peer Certificate** if desired.
5. Click **Submit**.

The new domain is imported via Git repository. Note that the domain is validated on import.

Changing Priority Order of Domains

Functions are executed depending on the order of Domain priority. Use this procedure to change the priority order of domains.

1. Navigate to **Automation > Automate > Explorer**. The default view is the Datastore.

2. Select the Domains you want to change the priority order for.
3. Click (**Configuration**), then (**Edit Priority Order of Domains**).
4. The list of Domains selected shows up. Note that you cannot change the priority of locked Domains and therefore locked Domains do not show up on the list.
5. Select one or more consecutive groups to move up or down to change their priority as required.
6. Click **Save**.

Creating a Namespace

1. Navigate to **Automation > Automate > Explorer**. The default view is the Datastore.
2. Navigate through the various Domains and Namespaces until you reach the desired location for your new Namespace.
3. Click (**Configuration**), then (**Add a New Namespace**).
4. Type in a unique **Name** and **Description**.
5. Click **Add**.

The new Namespace is created.

Creating a Class

1. Navigate to **Automation > Automate > Explorer**, navigate to the namespace you want to add a class to.
2. Click (**Configuration**), then (**Add a new Class**).
3. Type in a unique **Name** and **Description**.
4. If you want to use the schema from a class that has already been created, select it from the **Inherits From** dropdown. If the class that the new class inherits from changes, the new class will also change.
5. Click **Add**.

The new class is created and you can create a schema, add instances and methods.

Note:

For each class, create a schema if you did not choose to inherit from an existing class. The schema can include attributes, methods, assertions, and relationships.

Creating a Schema for a Class

This procedure shows you how to create a schema.

1. Navigate to **Automation > Automate > Explorer**, and click the class you want to define a schema for.
2. Click on the **Schema** tab.
3. Click (**Configuration**), then (**Edit selected Schema**).
4. Click (**Click to add a new field**) to create a new field.
5. Type in a **Name** for the new field.
6. From **Type**, select **Assertion**, **Attribute**, **Method**, **Relationship**, or **State**.
7. If applicable, select a **Data Type** and set a **Default Value**.
8. Type in a user friendly **Display Name** and **Description**.
9. Check **Sub** to enable the substitution syntax of **${}**. Uncheck it if you want to use that syntax as a regular string.
10. Fill in **Collect** and **Message** as required. **Collect** is used to roll up values from resolved relationships. For example, a relationship can resolve to a large object tree. Use **collect** to specify how to pull out data from those child objects into the current object. If you give **collect** a name value, it will store the method result in an attribute of the current object with that name.
11. **On Entry**, **On Exit**, **On Error**, **Max Retries**, and **Max Time** are fields used mostly for state machines. Leave blank if not applicable.
12. Click (**Add this entry**) to confirm the fields values.

13. For each new field, repeat steps 4 through 10.
14. When you have created all of the fields, click **Save**.

The class schema is created, and you can now add instances to it.

Note:

You may need to edit a class schema to reorder, add, edit, or remove a field. Classes define the order in which fields are processed and you may need to process some items before others.

Editing a Field in a Schema

This procedure describes how to edit schema fields.

1. Navigate to **Automation > Automate > Explorer**.
2. Click the class you want to define a schema for.
3. Click the **Schema** tab.
4. Click (**Configuration**), then (**Edit selected Schema**).
5. Make required changes to any of the definitions for the field.
6. To remove a field, click (**Click to delete this field from the schema**).
7. Click **Save** when you are finished editing the schema.

Once the schema is created, you can add instances and methods to the class.

Editing Schema Sequence

This procedure shows you how to change schema sequence.

1. Navigate to **Automation > Automate > Explorer**.
2. Click the class you want to change the schema sequence for.
3. Click the **Schema** tab.
4. Click (**Configuration**), then (**Edit Sequence**).
5. In the **Class Schema Sequencing** area, click the field you want to change the sequence for.
   - To move a field up in the order of resolving an instance, click (**Move selected field up**).
   - To move a field down in the order of resolving an instance, click (**Move selected field down**).
6. Click **Save** when you are finished editing the sequence.

Adding an Instance to a Class

This procedure shows you how to create an instance.

1. Navigate to **Automation > Automate > Explorer**.
2. Click the class you want to define a schema for.
3. Click the **Instances** tab.
4. Click (**Configuration**), then (**Add a new Instance**).
5. In the **Main Info** area, type in a **Name**, **Display Name** and **Description**.
6. In the **Fields** area, type in an appropriate value for each field, leave the field blank if no value is required, or use the default value.
7. Click **Add**.

Copying a Class or Instance

1. Navigate to **Automation > Automate > Explorer**. The default view is the Datastore.
2. Navigate through the various Domains and Namespaces until you reach the desired class or instance to copy.

3. Click (**Configuration**), then either (**Copy this Class**) or (**Copy this Instance**) depending on the object chosen.
4. Choose the target Domain in the **To Domain** drop-down menu.
5. The object retains the same path as the **From Domain** and overrides the class in **From Domain** if the **To Domain** has a high priority. You can also untick the **Copy to same path** option to specify a new Namespace.
6. Click **Add**.

Relationships

Relationships are used to connect to other instances in the **Automation Datastore**. Relationships are formed using URI syntax. The following can also be passed through a relationship:

- Use **#** to set the message to send to the item in the relationship.
- To pass an input to the method use **?** followed by the item to pass.
- If you want to use a substitution, the syntax is **$\{}** with the substitution located between the brackets.

Example	Explanation
`/Cloud/VM/Provisioning/Naming/Default#create`	This relationships uses the Default instance of the Naming class, which provides a means for other classes to name virtual machines. The relationship sends the `create` message to the class.
`/Cloud/VM/Provisioning/StateMachines/VMProvision_VM/AcquireMACAddress#$\{#ae_message}`	This relationships substitutes the message to send to the AcquireMACAddress instance of the VMProvision_VM class with the value in `ae_message`.
`/Cloud/VM/Retirement/Email/vm_retirement_emails?event=vm_retired`	Invokes the vm_retirement_emails instance of the Email class. Also sends the value `vm_retired` in the `event` attribute, which is used in the vm_retirement_emails method.
`/Service/Lifecycle/Retirement?service_id=$\{process#service_id}`	Invokes the Retirement instance of the Lifecycle class and send a replacement value in `process#service_id` to the `service_id` attribute.

Methods

Methods are pieces of code associated with a class or object to perform a task. Infrastructure Automation allows for Ruby methods or backing a method using an Ansible playbook. You can create your own methods or use relationships to link to pre-existing ones.

Infrastructure Automation ships with a core set of Ruby gems used by the Infrastructure Automation Rails Application. The Ruby gems in this set are subject to change. If you are calling gems using Automate that are no longer in this release, you can install them by using the `gem install` command.

While gems can be imported into automation methods using `require`, it is recommended that the authors of the automation methods clearly document the use of gems either in the core set or a custom set. It is the responsibility of the author of such custom automation to own the life cycle of any gem being referenced in those methods.

The Release Notes list Ruby gems that have been added, updated, or removed in the latest version of Infrastructure Automation.

Creating a Method

This procedure shows you how to create a method.

1. Navigate to **Automation > Automate > Explorer**, navigate to the class where you want to create a method.
2. Click the **Methods** tab.
3. Click (**Configuration**), (**Add a New Method**).
4. In the **Main Info** area, type in a **Name** and **Display Name**.
5. For **Location**, select **inline**. Once selected, you will be presented with a **Data** area in which to write or copy the script.
6. Click **Validate** to check the syntax.
7. Click **Add**.

Creating a Dynamic Content Dialog

The procedure describes the steps to create a dynamic content dialog.

1. Navigate to **Automation > Automate > Explorer**.
2. From the accordion menu, click **DOMAIN > Cloud > VM > Operations > Methods**.

Note:

DOMAIN must be a user-defined Domain and not the locked ManageIQ Domain. If necessary, you can copy the class from the ManageIQ domain into a custom domain.

This example uses the **Cloud** Namespace but can also use the **Infrastructure** namespace.

3. Click (**Configuration**), then (**Add a new Instance**).
4. In the **Main Info** area, enter **Name** = *dynamic_list*, replacing *dynamic_list* with an appropriate name for the method.
5. Enter a **Display Name** and **Description**.
6. In the **Fields** area, enter **Value** = *dynamic_list*. Leave the other fields blank or use the default values.
7. Click **Add**.
8. Navigate to **Methods** tab.
9. In the **Main Info** area, enter **Name** = *dynamic_list* and populate the **Data** section with the example automate method below.
10. Click **Add**.
11. Set the automate entry point for the dialog control; use the new instance created in step four. You can create a new domain and copy the method to that domain.

```
#  Automate Method

dialog_field = $evm.object

# sort_by: value / description / none
dialog_field["sort_by"] = "value"

# sort_order: ascending / descending
#dialog_field["sort_order"] = "ascending"

# data_type: string / integer
dialog_field["data_type"] = "integer"

# required: true / false
# dialog_field["required"] = "true"

dialog_field["values"] = {1 => "one", 2 => "two", 10 => "ten", 50 => "fifty"}
dialog_field["default_value"] = 2
```

Creating a Playbook Automate Method

Infrastructure Automation can choose an Ansible playbook from a repository and execute it as a method. Each playbook method can take additional input parameters specified by the user.

Important:

- You must first sync your playbook repositories before using them to create a method. See Adding a Playbook Repository in *Managing Providers* for information on initial playbook repository set-up.
- Using Ansible playbooks to populate dynamic dialog fields is not recommended due to delay times caused by the overhead of interaction between systems.
- Only users with administrator privileges can run a service dialog based on a playbook automate method.

To create a playbook automate method:

1. Navigate to **Automation > Automate > Explorer**, then click on a domain under **Datastore**.
2. Under a namespace, select the class for which you want to create a new method.
3. Click the **Methods** tab.
4. Click (**Configuration**) then, (**Add a New Method**).
5. In the **Main Info** area, select **Playbook** from the **Type** dropdown menu.
6. Provide a **Name** and **Display Name**.
7. Select a playbook **Repository** from the list.
    1. Choose a **Playbook** to use.

2. Select the **Machine Credential** the playbook will use when it runs.
3. Select the **Vault Credential** to use.
4. From the **Cloud Type** list, select a cloud provider.
5. Choose the **Cloud Credential** that corresponds to the selected cloud type.

8. Specify the **Hosts** on which the playbook will run. Choose **Localhost** or provide unique values in the **Specify host values** field.
9. Set the **Max TTL** in minutes. The Time To Live (TTL) field allows you to set the maximum execution time for the playbook to run.
10. Select when to receive **Logging Output** from the options in the drop-down menu.
11. Use the **Escalate Privilege** toggle switch to enable user privilege escalation if credentials are called for during the playbook run.
12. Choose a **Verbosity** value to set the debug level for playbook execution.
13. Add required **Input Parameters** using the fields and values available. Click the + to add additional input parameters.

    Note:

    Input parameters become extra vars, with substitution enabled. This overcomes the lack of a dialog which would normally allow for the input of additional information. For more information on extra vars, see the Ansible documentation.
14. Click **Add** when finished.

Once created, the domain including your playbook method can be exported to appliances in your testing or production environments or imported in appliances in multiple regions.

Important:

To import a domain with a playbook method, you must have an existing Ansible playbook on the destination environment with the same name or the import will fail.

See Exporting All Datastore Classes for information.

Passing variables between successive playbook methods

Automate designers can pass variables between successive Ansible playbook methods in a state machine using Infrastructure Automation.

To pass variables between Ansible playbook methods, use the `set_stats` module in your playbooks.

Example.

```
---
# This playbook prints a simple debug message
- name: Echo Hello, Fred and Wilma
  hosts: localhost

  tasks:
  - name: "playbook1 - testing input vars"
    set_stats:
      data:
        pb_var1a: ""
        pb_var1b: ""

  - debug: msg="Hello,  and "
```

For more information on the `set_stats` module, see the Ansible documentation.

Expression Methods

Infrastructure Automation additionally provides support for Expression Methods, that allow you to use advance search filters as Automate Methods, substituting the user input from Automate Objects. Expression methods have several distinct advantages, including: running directly in the worker appliance; removing the overhead of forking a DRb process to run the Automate Methods; no Ruby code required; and prebuilt for Dynamic Dialogs.

Input Parameters

Expression methods allow for substitution of user input through input parameters,

Input Parameter	Explanation
arg	The argument used in the expression. Each argument should employ the prefix **arg**. Example: **arg1**: the first argument in the expression; **arg2**: the second argument in the expression; **argn**: the nth argument in the expression.

Optional Input Parameters

Note:

If attributes and distinct are not specified we try to store the result in a variable called values with a hash consisting of id and name. This makes it compatible with our existing dynamic dialog result set.

Optional Input Parameter	Explanation
attributes	A comma delimited list of attributes to select from the resultant objects. This should me marked as an Array Type in the Input Parameters field.
distinct	A comma delimited list of attributes which are distinct in the resultant objects.This should me marked as an Array Type in the Input Parameters field.
result_obj	The object where the result data should be stored. (default: current object)
result_attr	The name of the attribute which stores the result. (default: values)
result_type	The result type hash or array (default: dialog_hash which matches to our dynamic dialog hash. Valid values are 'hash', 'dialog_hash', 'array', 'simple'
on_empty	The method behavior when the search returns an empty list.
error	Abort. (default: error)
default	The default value in case the result is empty and you select warn.

Creating an Expression Method

Expression methods allow you to use advance search filters as automate methods, substituting user input at runtime, and making them ideal for dynamic dialogs.

To create an expression method:

1. Navigate to **Automation > Automate > Explorer**, then click on a domain under **Datastore**.
2. Under a namespace, select the class for which you want to create a new method.
3. Click the **Methods** tab.
4. Click (**Configuration**) then, (**Add a New Method**).
5. In the **Main Info** area, select **expression** from the drop-down menu.
6. Provide a **Name** and **Display Name**.
7. Select an **Expression Object** from the drop-down menu.
8. In the **Expression** editor, create the expression by setting the controls and values used at runtime:
    1. Using the drop-down menu, select the value to use. Based on your selection, choose or input additional values from the drop-down menus or text fields that appear.
    2. In the **Contains** field, input a value or click **User will input the value**.
    3. Click to complete the expression.
9. Add **Input Parameters** for each of the user input fields required.
    1. Click to add a new parameter.
    2. Provide a **Name**, **Default Value** and select a **Data Type** for each parameter.
    3. Click to add the parameter.

       Note:

       If **User will input the value** is checked, arguments for each input parameter names using the prefix "arg".

       For example, if there are 3 fields then the input parameter names should be arg1, arg2, and arg3. If there are two runtime parameters arg1 and arg2 must be defined in the input parameters. In the default value for these fields values can be substituted from other objects in the Automate Workspace.
10. Click **Add**.

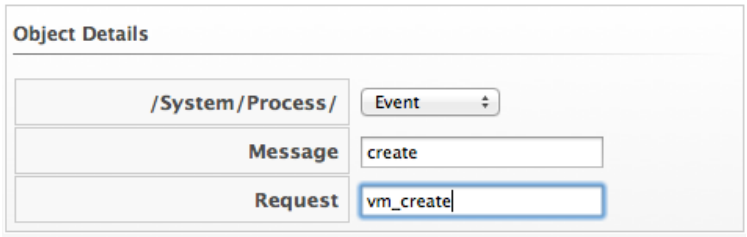
Simulation

After your model is designed, use the simulate page to test it. It allows you to see the results in tree and XML view.

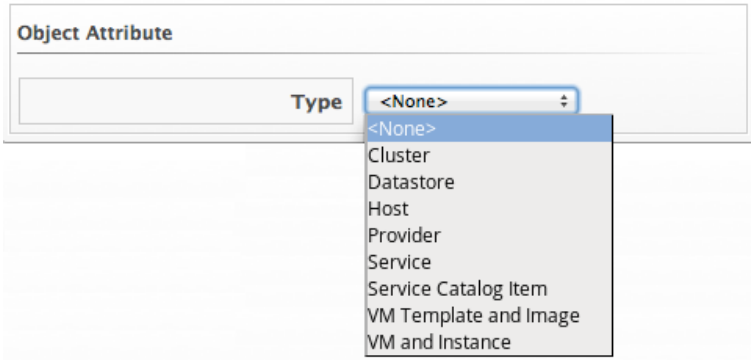
Simulating an Automate Model

This procedure shows you how to simulate an automate model.

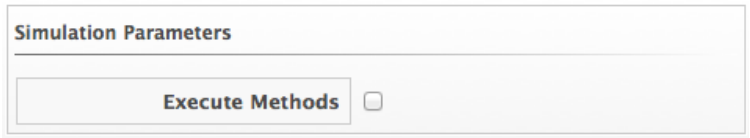
1. Navigate to **Automation > Automate > Simulation**.
2. In **Object Details**, select a type of object from `/System/Process/` that will initiate the model. The **Message** should be `create`. Type in the name of the **Request** where you are starting from.



3. Select the **Type** of item you want to run the simulation on. Then, select a specific one to use as the example.



4. Check **Execute Methods** if you want to perform the model and not just simulate it.



5. Type in the **Attribute/Value Pairs** fields if applicable.
6. Click **Submit**.

Click on the **Tree View** or **XML View** tabs to see results.

Importing, Exporting, and Resetting the Datastore

The **Automate Model** can be exported and imported as a YAML file. Infrastructure Automation allows you to back up your model by export. Red Hat may provide you with new or updated classes, and provides an import function for either a class or the entire model. Finally, you can reset the datastore to its default. Always be sure to export the current datastore before importing or resetting.

Note:

The datastores you are exporting or importing between must use the same Infrastructure Automation version.

Exporting All Datastore Classes

This procedure shows you how export datastore classes as a YAML file.

1. Navigate to **Automation > Automate > Import/Export**.
2. Click (**Export all Datastore classes and instances to a file**).
3. Follow your browser's prompts to save the file.

The datastore is zipped and exported as a `YAML` file.

Importing Datastore Classes

This procedure shows you how to import datastore classes.

1. Navigate to **Automation > Automate > Import/Export**.
2. Export the datastore so that you have a backup.
3. Click **Choose file** to navigate to the location of the **.zip* file to import.
4. Click **Upload**.
5. After the file has uploaded, specify the following using the dropdown menus:
    - The domain to import into
    - The domain to import from
    - All namespaces you wish to import
6. Click **Commit** to finish importing the selected domain.

The datastore is imported from the uploaded **`YAML`** files.

Resetting Datastores to the Default

This procedure shows you how reset the datastores to the default.

1. Navigate to **Automation > Automate > Import/Export**.
2. Export the datastore so that you have a backup.
3. Click (**Reset all components in the built-in domains**).
4. Click **OK**.

Invoking Automate

Automate Examples

This chapter describes the ways to invoke an Automate workflow. Automation can be initiated through an alert, an event, a Infrastructure Automation application, or a custom button. The same automation process can be re-used across more than one of these. For example, using automation to remove orphaned virtual machines and instances could be initiated by:

- An administrator request from the Infrastructure Automation console (from a custom button)
- An alert indicating the datastore has less than 20% free-space
- A virtual machine or instance unregistered event is detected

All invocations of an automate model must enter through the **`/System/Process`** namespace.

Invoking Automate using a Custom Button

Invoke an Automate model by mapping an Ansible playbook or instance from the **`/System/Process/Request`** class to a custom button. Before creating the button, you need to have an Ansible playbook service catalog item or an instance in the **`/System/Process/Request`** class to map to it and a button group to assign it to.

Create buttons for a cluster, host, datastore, provider, virtual machines or cloud instances. When the button is clicked, the model or playbook will be invoked for the selected item. For each of these, you can have up to 15 buttons.

Creating a Custom Button Group

This procedure shows you how to create a custom button group.

1. Browse to menu: **Automation > Automate > Customization**.
2. Click the **Buttons** accordion.
3. From the **Object Types** tree, select the type of object you want to create the button group for.
4. Click **Configuration**, **Add a new Button Group**.
5. Type in a **Button Group Text** and **Button Group Hover Text**, and select the **Button Group Image** that you want to use.
6. If custom buttons have already been created, assign them to the button group. If not, see Creating a Custom Button.
7. Click **Add**.

The button group will show in the object type you added the button to.

Creating a Custom Button

This procedure shows you how to create a custom button.

Note:

Custom buttons can be migrated to other Infrastructure Automation appliances. See Migrating Custom Buttons for guidance on migrating custom buttons to a new Infrastructure Automation appliance.

1. Browse to menu: **Automation > Automate > Customization**.
2. Click the **Buttons** accordion.
3. From the **Object Types** tree, select the type of object you want to create the button for.
4. Click **Unassigned Buttons**.
5. Click **Configuration**, then **Add a new Button**.

   Note:

   If **Add a new Button** is not available, that means you have not created a button group for that object. To continue, create a button group first. See Creating a Custom Button.
6. Under the **Options** tab:
   1. Select the **Button** type from the list.
   2. Enter button **Text** and **Hover Text**, and select the **Icon** and **Icon Color** to use.
   3. Select a **Dialog** if applicable.
   4. Check **Open URL** to open a browser window for the custom URL returned when the button is executed.
   5. Choose a **Display For** option for the button.
   6. Select a **Submit** parameter to choose how to submit objects to automate. Selecting *Submit All* will pass all objects at once when the button is executed, while choosing *One by one* will run execute the button action each time per object.
7. Under the **Advanced** tab:
   1. Set button **Enablement**. Click **Define Expression** to access the expression editor tool. Enter **Disabled Button Text** to display when the custom button is disabled.
   2. Use **Visibility** to determine button appearance based on a custom expression. Click **Define Expression** to access the expression editor tool.

      Note:

      For more about setting visibility and enablement for a custom button, see Filtering.
   3. In **Object Details**, select **Request** from the **`/System/Process/`** dropdown. By default, the message is **`create`**. Do not change it.
   4. Enter a **Request** name for the **`/System/Process/Request`** instance.
   5. Enter the **Attribute/Value Pairs** fields if applicable.
   6. Set **Role Access**. Selecting *<By Role>* will display available roles. Check applicable roles.
8. Click **Add** when you have confirmed that the button accomplishes the task you want.

The button will show in the object type you added the button to.

Please note that there are two ways of creating custom buttons, and creation through the “Automation > Automate > Generic Objects” screen will necessitate adding an attribute/value pair that is not required when adding through the “Automation > Customization > Buttons” screen.

Creating an Ansible Playbook Button

Infrastructure Automation includes an option to create an Ansible Playbook custom button. This feature allows users to select a playbook to run as well as an inventory target to run it against. An Ansible playbook type button can be defined for any object type available.

Note:

An Ansible Playbook catalog item must exist in order to create an Ansible Playbook custom button. For more information, see Creating an Ansible Playbook Service Catalog Item in the Provisioning Virtual Machines and Hosts guide.

1. Navigate to **Automation > Automate > Customization**.
2. Click the **Buttons** accordion.
3. From the **Object Types** tree, select the type of object you want to create the button for.
4. Click **Unassigned Buttons**.
5. Click (**Configuration**), then (**Add a new Button**).

   Note:

   If (**Add a new Button**) is not available, that means you have not created a button group for that object. To continue, see Creating a Custom Button Group.
6. Select **Ansible Playbook** from the **Button Type** drop-down menu.
7. From the **Playbook Catalog Item** choose a playbook-backed catalog item to run.
8. Choose a host from the **Inventory** against which to run the playbook. If **Specific Hosts** is selected, input the IP address or DNS names for each host in the text field, separating each with a comma.

   Note:

   Infrastructure Automation supports two configurations for host value input:

   - To allow user-provided host values, set the custom button to **Specific Hosts** and leave the associated text field blank.
   - To use admin-specified host values, remove the **Hosts** field when creating the dialog the service uses. In this configuration, the field will not appear to users. See Service Dialogs for information on generating a service dialog.
9. Type in a **Text** and **Hover Text**, and select the **Icon** you want to use.
10. Select an **Icon Color** from the color selection palette that pops up.
11. Check **Open URL** to open a browser window for the custom URL returned when the playbook is run.
12. Select display options for the Ansible Playbook button from the **Display for** drop-down menu. Choose for the button to display in the list view, for single entities, or both.
13. Choose how to submit objects to automate by selecting an option from the **Submit** drop-down menu. Selecting **Submit all** will pass all objects at once when the playbook is executed, while choosing **One by one** will run the the playbook each time per object.
14. Click **Add** when you have confirmed that the button accomplishes the task you want.

Editing a Custom Button

This procedure shows you how to edit a custom button.

1. Navigate to **Automation > Automate > Customization**.
2. From the **Object Types** dropdown, select the type of object you want to edit the button for.
3. Click the button you want to edit.
4. Click (**Configuration**), (**Edit this Button**).
5. Modify as required.
6. Click **Save**.

Deleting a Custom Button

This procedure shows you how to delete a custom button.

1. Navigate to **Automation > Automate > Customization**, then select the **Buttons** accordion.
2. From the **Object Type** tree in the accordion menu, select the type of object you want to remove the button from.
3. Click (**Configuration**). then click (**Remove this button**).
4. Click **OK** to confirm that you want to delete this button.

Setting Enablement and Visibility for Custom Buttons

Infrastructure Automation adds methods for evaluating an expression to determine whether a custom button is visible and enabled. Each method has a target object, for example, a virtual machine or host, and expressions can set a custom button to visible, hidden, or disabled.

Note:

Filtering works on single objects and is not applicable to lists.

To apply filtering actions to a custom button:

1. Navigate to **Automation > Automate > Customization**.
2. Click the **Buttons** accordion. Select the custom button to use.
3. Click (**Configuration**), then (**Edit this Button**).
4. Click the **Advanced** tab.
5. To set enablement filtering on a custom button:
    1. Under **Enablement**, click on **Define Expression**.
    2. Create a visibility expression using the expression editor tools.
    3. Click (**Confirm**) when finished defining the expression.
    4. Provide **Disabled Button Text** in the field.
6. To set visibility filtering on a custom button:
    1. Under **Visibility**, click on **Define Expression**.
    2. Create a visibility expression using the expression editor tools.
    3. Click (**Confirm**) when finished defining the expression.
7. Click **Save**.

Using a Custom Button

This procedure shows you how to use custom buttons to invoke a cluster, host, datastore, provider, virtual machine or instance.

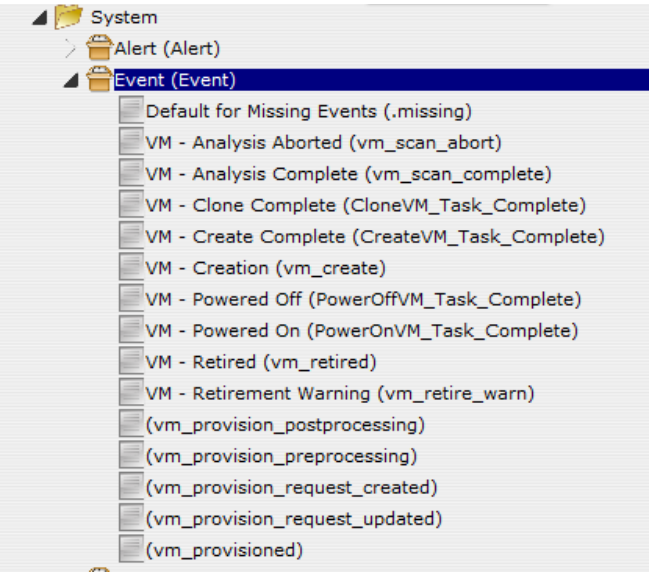
1. Go to the page for the item that you created a button for.
2. Click the custom button group from the taskbar, and then your custom button.

The automate model is invoked for the specified item.

Initiating Automate from an Event

You can also use a Infrastructure Automation Policy Event to initiate automation. You can either use the provided Raise Automation Event action or create a custom automation action. The first case will start in the **`/System/Process`** class, but then go to the Event that initiated the Automate model in the **`/System/Process/Event Class`**. If you create your own custom action, it will start from the **`/System/Process`** class and then go to the **`/System/Process/Request Class`** instead.

For example, suppose that you always want the same Automate model to occur when a virtual machine is created. You would use the Raise Automation Event Action. There are instances in the **`/System/Process/Event Class`** for the following Events that you can select as part of a Policy:



Creating a Policy for Automate

This procedure shows you how to create a policy for automate.

1. Navigate to **Control > Explorer**.
2. Click the **Policies** accordion, and select **Control Policies**.
3. Select **Vm Control Policies**.
4. Click (**Configuration**), then (**Add a New Control Vm Policy**).
5. Type in a **Description**.
6. Uncheck **Active** if you do not want this policy processed even when assigned to a resource.
7. Click **Add**. You are brought to the page where you add conditions and events to your new policy.
8. Click (**Configuration**), then (**Edit this Policy's Event assignments**).
    - Check the events you want to use to send to an **Automate Model**.
    - Click **Save**.
    - From the **Events** area, click on the **Description of the Event** you want to assign an action to.
    - Click (**Edit Actions for this Policy Event**).
9. Select **Raise Automation Event**, and click (**Move selected Actions into this Event**).
10. Click **Save**.

You can now assign this policy to a **Policy Profile**. Then, assign the policy profile to the virtual machines. Every time this event happens on the virtual machine the appropriate Automate Model will be initiated.

Note:

If you want the policy to initiate an Automate Model from the `/System/Process/Request` class, then you can create your own custom action. Be sure to have an instance in the `/System/Process/Request` class for it to map to.

Creating a Custom Automate Action

This procedure shows you how to create a custom Automate action.

1. Navigate to **Control > Explorer** accordion.
2. Click the **Actions** accordion.
3. Click (**Configuration**), then (**Add a new Action**).
4. Type in a **Description** for the Action.

5. Select **Invoke a Custom Automation** from **Action Type**.
6. In **Custom Automation**,
    - For **Message**, type `create`.
    - For **Request**, type in the name of the instance of the `/System/Process/Request Class` in the second.
7. Type in the **Attribute/Value Pairs** fields if applicable.
8. Click **Add**.

The action is created and can be added to a policy.

Objects

Virtual Machine Properties

When using these items in a method, prefix them with `vm.`. For example: `vm.storage_id`.

Friendly Name or Description	Raw Column Name
Allocated Disk Storage	allocated_disk_storage
Autostart	autostart
Blackbox Exists	blackbox_exists
Blackbox Validated	blackbox_validated
Boot Time	boot_time
Busy	busy
Cluster	ems_cluster_name
Configuration XML	config_xml
Connection State	connection_state
CPU Affinity	cpu_affinity
CPU Limit	cpu_limit
CPU Reserve	cpu_reserve
CPU Reserve Expand	cpu_reserve_expand
CPU Shares	cpu_shares
CPU Shares Level	cpu_shares_level
Created on Time	ems_created_on
Currently Used Space	used_storage_by_state
Datastore Path	v_datastore_path
Date Created	created_on
Date Updated	updated_on
Description	description
Ems	ems_id
Evm Owner	evm_owner_id
Evm Owner Email	evm_owner_email
Evm Owner Name	evm_owner_name
Infrastructure Automation Unique ID (Guid)	guid
Format	format
Host	host_id
Host Name	host_name
Id	id
Is a Template	v_is_a_template
Last Analysis Attempt On	last_scan_attempt_on
Last Analysis Time	last_scan_on
Last Compliance Status	last_compliance_status
Last Compliance Timestamp	last_compliance_timestamp
Last Perf Capture On	last_perf_capture_on
Last Sync Time	last_sync_on
Location	location
Memory Limit	memory_limit

Friendly Name or Description	Raw Column Name
Memory Reserve	memory_reserve
Memory Reserve Expand	memory_reserve_expand
Memory Shares	memory_shares
Memory Shares Level	memory_shares_level
Name	name
OS Name	os_image_name
Owner	owner
Paravirtualization	paravirtualization
Parent Cluster	v_owning_cluster
Parent Datacenter	v_owning_datacenter
Parent Folder (Hosts & Clusters)	v_owning_folder
Parent Folder (VMs & Templates)	v_owning_blue_folder
Parent Folder Path (Hosts & Clusters)	v_owning_folder_path
Parent Folder Path (VMs & Templates)	v_owning_blue_folder_path
Parent Host Platform	v_host_vmm_product
Parent Resource Pool	v_owning_resource_pool
Pct Free Disk	v_pct_free_disk_space
Platform	platform
Power State	power_state
Previous State	previous_state
Registered	registered
Reserved	reserved
Retired	retired
Retirement	retirement
Retires On	retires_on
Service	service_id
Smart	smart
Standby Action	standby_action
State Changed On	state_changed_on
Storage	storage_id
Storage Name	storage_name
Template	template
Thin Provisioned	thin_provisioned
Tools Status	tools_status
Total Provisioned Space	provisioned_storage
Total Snapshots	v_total_snapshots
Total Used Disk Space	used_disk_storage
Uid Ems	uid_ems
Uncommitted Space	uncommitted_storage
Used Storage	used_storage
V Pct Used Disk Space	v_pct_used_disk_space
VDI Available	vdi_available
VDI Connection DNS Name	vdi_connection_dns_name
VDI Connection Logon Server	vdi_connection_logon_server
VDI Connection Name	vdi_connection_name
VDI Connection Remote IP Address	vdi_connection_remote_ip_address
VDI Connection Session Name	vdi_connection_session_name
VDI Connection Session Type	vdi_connection_session_type
VDI Connection URL	vdi_connection_url
VDI Endpoint IP Address	vdi_endpoint_ip_address
VDI Endpoint MAC Address	vdi_endpoint_mac_address
VDI Endpoint Name	vdi_endpoint_name
VDI Endpoint Type	vdi_endpoint_type
VDI User Appdata	vdi_user_appdata
VDI User DNS Domain	vdi_user_dns_domain
VDI User Domain	vdi_user_domain
VDI User Home Drive	vdi_user_home_drive

Friendly Name or Description	Raw Column Name
VDI User Home Path	vdi_user_home_path
VDI User Home Share	vdi_user_home_share
VDI User Logon Time	vdi_user_logon_time
VDI User Name	vdi_user_name
Vendor	vendor
Version	version
VMsafe Agent Address	vmsafe_agent_address
VMsafe Agent Port	vmsafe_agent_port
VMsafe Enable	vmsafe_enable
VMsafe Fail Open	vmsafe_fail_open
VMsafe Immutable VM	vmsafe_immutable_vm
VMsafe Timeout (ms)	vmsafe_timeout_ms

Virtual Machine Properties

Methods for use in Ruby Scripts

To use one of these in one of your own Ruby methods, use the syntax of **`vm.method`**. For example, to reboot the guest operating system, use **`vm.rebootGuest`**.

Method	Description
start	Start Virtual Machine container.
stop	Stop Virtual Machine container.
suspend	Suspend Virtual Machine container.
unregister	Unregister Virtual Machine.
collect_running_processes	Collect the running processes from a started Virtual Machine.
shutdownGuest	Shutdown the guest operating system of the VM container. Requires VMware tools (or vendors tools) installed on the guest.
standbyGuest	Put the guest operating system into standby. Requires VMware tools (or vendors tools) installed on the guest.
rebootGuest	Reboot the guest operating system. Requires VMware tools (or vendors tools) installed on the guest.

Host Properties

When using these items in a method, prefix them with host, such as **`host.ems_id`**.

Friendly Name or Description	Raw Column Name
All Enabled Ports	all_enabled_ports
Annotation	v_annotation
Authentication Status	Authentication_status
Connection State	connection_state
CPU usage MHz rate average over time period	cpu_usagemhz_rate_average_avg_over_time_period
CPU usage MHz rate high over time period	cpu_usagemhz_rate_average_high_over_time_period
CPU usage MHz rate low over time period	cpu_usagemhz_rate_average_low_over_time_period
Custom Attribute 1	custom_1
Custom Attribute 2	custom_2
Custom Attribute 3	custom_3
Custom Attribute 4	custom_4
Custom Attribute 5	custom_5
Custom Attribute 6	custom_6
Custom Attribute 7	custom_7
Custom Attribute 8	custom_8
Custom Attribute 9	custom_9
Date Created	created_on
Derived memory usage average over time period	derived_memory_used_avg_over_time_period
Derived memory usage high over time period	derived_memory_used_high_over_time_period
Derived memory usage low over time period	derived_memory_used_low_over_time_period
Ems	ems_id
Enabled Inbound Ports	enabled_inbound_ports
Enabled Outbound Ports	enabled_outbound_ports
Enabled Run Level 0 Services	enabled_run_level_0_services

Friendly Name or Description	Raw Column Name
Enabled Run Level 1 Services	enabled_run_level_1_services
Enabled Run Level 2 Services	enabled_run_level_2_services
Enabled Run Level 3 Services	enabled_run_level_3_services
Enabled Run Level 4 Services	enabled_run_level_4_services
Enabled Run Level 5 Services	enabled_run_level_5_services
Enabled Run Level 6 Services	enabled_run_level_6_services
Enabled TCP Inbound Ports	enabled_tcp_inbound_ports
Enabled TCP Outbound Ports	enabled_tcp_outbound_ports
Enabled UDP Inbound Ports	enabled_udp_inbound_ports
Enabled UDP Outbound Ports	enabled_udp_outbound_ports
EVM Unique ID (Guid)	guid
Hostname	hostname
Id	id
IP Address	ipaddress
Last Compliance Status	last_compliance_status
Last Compliance Timestamp	last_compliance_timestamp
Last Perf Capture On	last_perf_capture_on
Last Analysis Time	last_scan_on
Name	name
OS Name	os_image_name
Platform	platform
Power State	power_state
Region Description	region_description
Region Number	region_number
Reserved	reserved
Service Names	service_names
Settings	settings
Smart	smart
SSH Root Access	ssh_permit_root_login
Uid Ems	uid_ems
Date Updated	updated_on
User Assigned Os	user_assigned_os
Parent Cluster	v_owning_cluster
Parent Datacenter	v_owning_datacenter
Parent Folder (Hosts & Clusters)	v_owning_folder
Total Datastores	v_total_storages
Total VMs	v_total_vms
VMM Build Number	vmm_buildnumber
VMM Platform	vmm_product
VMM Vendor	vmm_vendor
VMM Version	vmm_version

Provider Properties

When using these items in a method, prefix them with **`ems`**, such as **`ems.ems_id`**.

Friendly Name or Description	Raw Column Name
Aggregate VM CPUs	aggregate_vm_cpus
Aggregate VM Memory	aggregate_vm_memory
CPU Ratio	v_cpu_vr_ratio
CPU Usage MHZ Rate Average High Over Time Period	cpu_usagemhz_rate_average_high_over_time_period”
CPU Usage MHZ Rate Average Low Over Time Period	cpu_usagemhz_rate_average_low_over_time_period
CPU Usage MHZ Rate Average Over Time Period	cpu_usagemhz_rate_average_avg_over_time_period
Date Created	created_on
Date Updated	updated_on
Derived Memory Usage Rate Average High Over Time Period	derived_memory_used_high_over_time_period
Derived Memory Usage Rate Average Low Time Period	derived_memory_used_low_over_time_period
Derived Memory Usage Rate Average Over Time Period	derived_memory_used_avg_over_time_period

Friendly Name or Description	Raw Column Name
Distributed Resource Scheduler Automation Level	drs_automation_level
Distributed Resource Scheduler Enabled	drs_enabled
Distributed Resource Scheduler Migration Threshold	drs_migration_threshold
EMS ID	ems_id
EVM Zone	zone_name
High-Availability Admission Control	ha_admit_control
High-Availability Enabled	ha_enabled
High-Availability Max Failures	ha_max_failures
Id	id
Last Performance Data Captured	last_perf_capture_on
Last Smart State Analysis	last_scan_on
Memory Ratio	v_ram_vr_ratio
Name	name
Parent Datacenter	v_parent_datacenter
Qualified Description	v_qualified_desc
Region Description	region_description
Region Number	region_number
Reserved	reserved
Total CPU Speed	aggregate_cpu_speed
Total Hosts	total_hosts
Total Memory	aggregate_memory
Total Number of Logical CPUs	aggregate_logical_cpus
Total Number of Physical CPUs	aggregate_physical_cpus
Total Vms	total_vms
Unique Identifier	uid_ems

Storage Properties

When using these items in a method, prefix them with **`storage`**, such as **`storage.name`**.

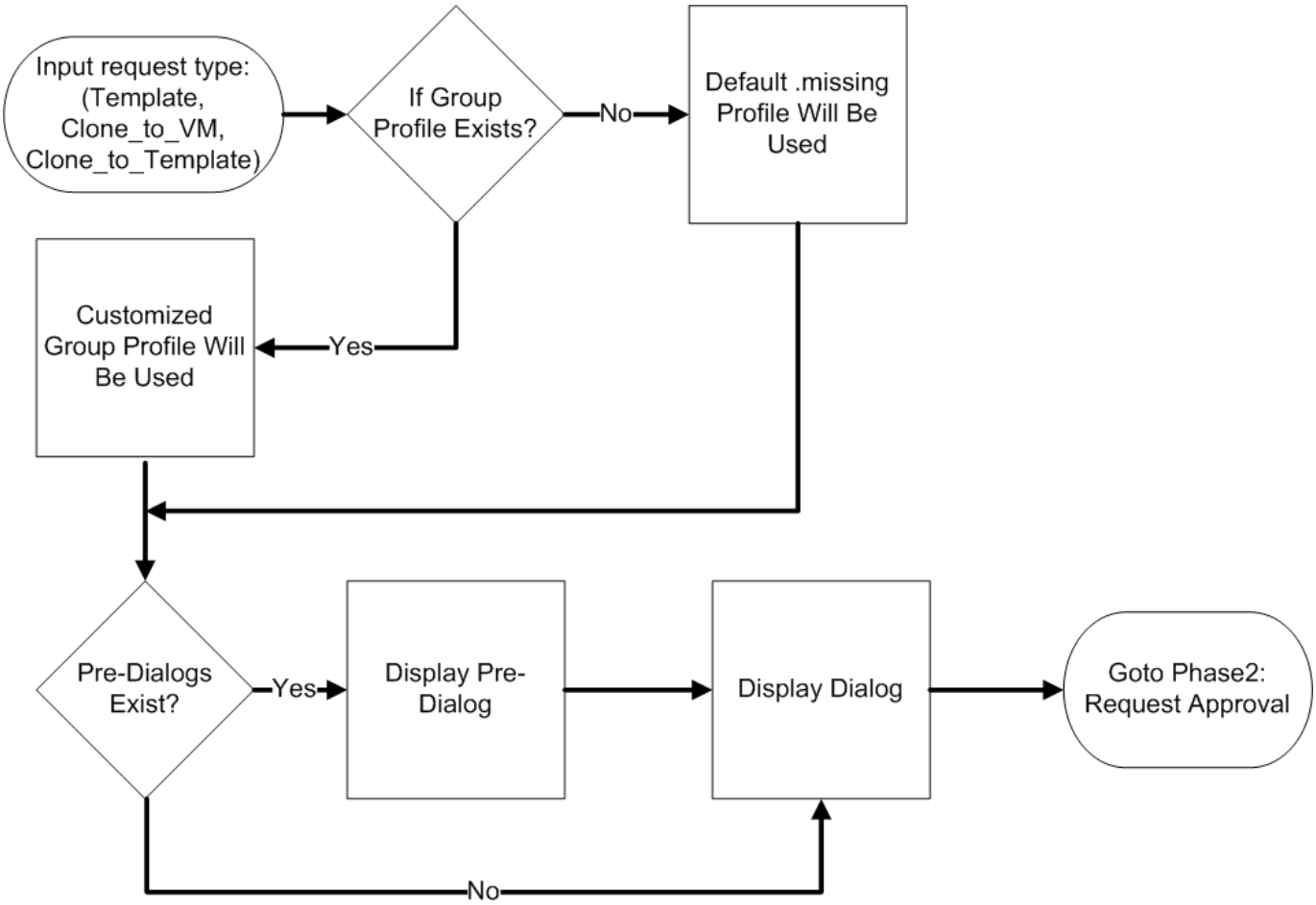
Friendly Name or Description	Raw Column Name
Date Created	created_on
Date Updated	updated_on
Disk Files Percent of Used	v_disk_percent_of_used
Free Space	free_space
Free Space Percent of Total	v_free_space_percent_of_total
Id	id
Last Analysis Time	last_scan_on
Last Perf Capture On	last_perf_capture_on
Location	location
Multiple Host Access	multiplehostaccess
Name	name
Non-VM Files Percent of Used	v_debris_percent_of_used
Other VM Files Percent of Used	v_vm_misc_percent_of_used
Provisioned Space Percent of Total	v_provisioned_percent_of_total
Reserved	reserved
Size of Non-VM Files	v_total_debris_size
Size of Other VM Files	v_total_vm_misc_size
Size of VM Memory Files	v_total_vm_ram_size
Size of VM Snapshot Files	v_total_snapshot_size
Snapshot Files Percent of Used	v_snapshot_percent_of_used
Store Type	store_type
Total Hosts	v_total_hosts
Total Managed Registered Vms	total_managed_registered_vms
Total Managed Unregistered Vms	total_managed_unregistered_vms
Total Provisioned Space	v_total_provisioned
Total Space	total_space
Total Unmanaged Vms	total_unmanaged_vms

Friendly Name or Description	Raw Column Name
Total VMs	v_total_vms
Uncommitted	uncommitted
Used Space	v_used_space
Used Space Percent of Total	v_used_space_percent_of_total
VM Memory Files Percent of Used	v_memory_percent_of_used

Storage Properties

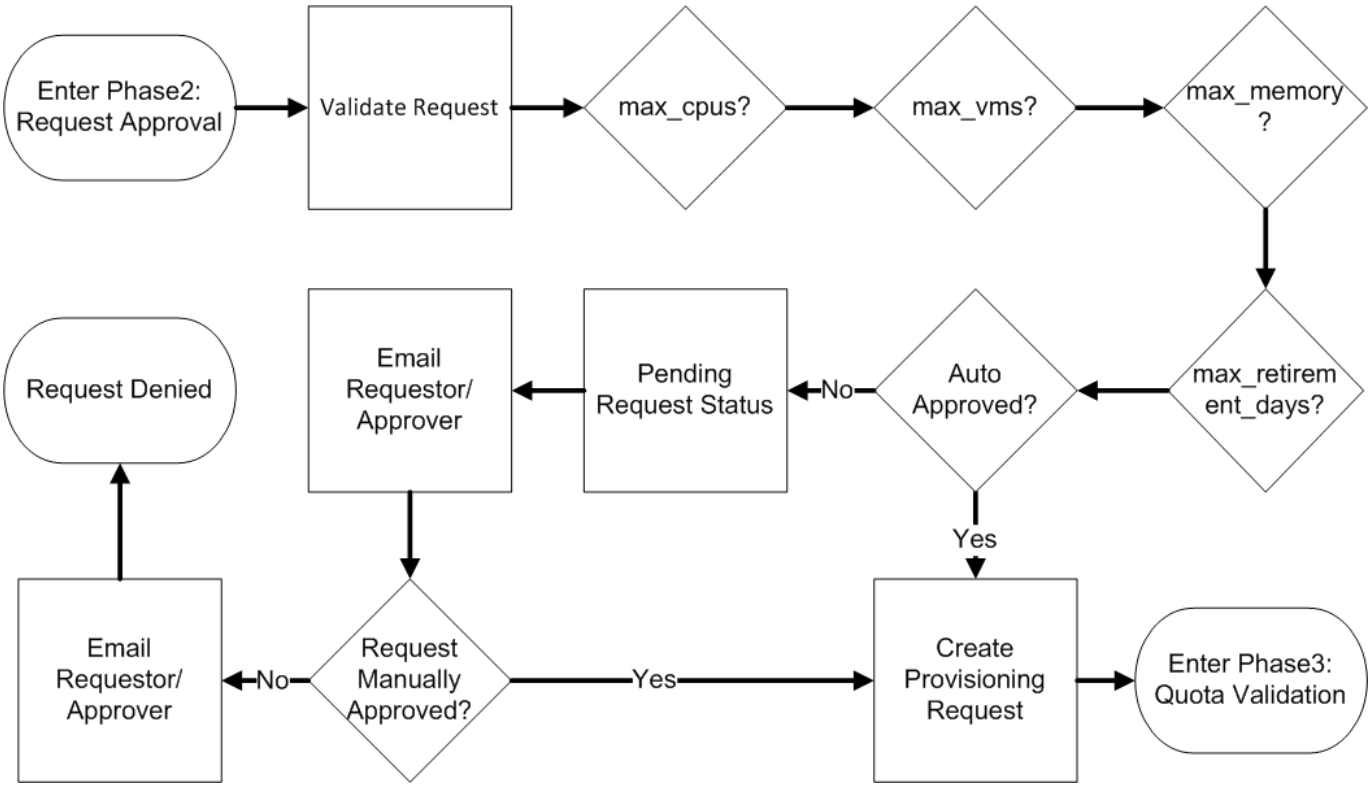
FAQs and Flows

Phase 1: Create Provision Request



Question	Answer
Where do I create a new provisioning profile based on a users LDAP group?	Navigate to [`VM / Provisioning / Profile`] of either the **Cloud** or **Infrastructure** namespace in your domain.
Where can I specify a pre-dialog to present to a Requester in their LDAP group?	Custom pre-dialogs can be defined in [` VM / Provisioning /
Profile / `\] of either the **Cloud** or **Infrastructure** namespace in your domain.	
I would like to customize our dialogs. Where are all the dialogs kept?	All dialogs are located on each Infrastructure Automation appliance in the `/var/www/miq/vmdb/db/fixtures` directory.
What happens if I do not specify any profiles for provisioning?	Infrastructure Automation searches for a matching LDAP group in the [`VM / Provisioning / Profile`] class of either the **Cloud** or **Infrastructure** namespace in your domain. If an LDAP profile is NOT found then Infrastructure Automation will use the `missing class instance`.

Phase 2: Request Approval



Question	Answer
Where can I specify auto-approval values on a per virtual machine template basis?	Tags can be assigned to templates in the form of [**prov_max_vms**, **prov_max_cpus**, **prov_max_memory**, **prov_max_retirement_days**].
Where can I modify the default Auto-Approval values?	These values can be set in the [`Service / Provisioning / StateMachines /
ServiceProvisionRequestApproval / Default`] class instance in your domain.	
How can I customize the email that is sent when a request is approved?	The Request Approved email message can be modified in [**`VM / Provisioning / Email / MiqProvisionRequest_Approved`**] in either the **Cloud** or **Infrastructure** namespace of your domain.
How can I customize the email that is sent when a request is denied?	The Request Denied email message can be modified in [**`VM / Provisioning / Email / MiqProvisionRequest_Denied`**] in either the **Cloud** or **Infrastructure** namespace of your domain.
How can I customize the email that is sent when a request is not Auto-approved?	The Request Pending email message can be modified in [**`VM / Provisioning / Email / MiqProvisionRequest_Denied`**] in either the **Cloud** or **Infrastructure** namespace of your domain.
If a Request Approval requires manual approval, how does an Approver approve the request?	Log into Infrastructure Automation as an approver/admin and navigate to **Virtual Machines > Requests** and then click on the request.

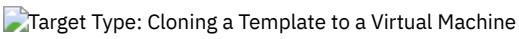
Phase 3: Quota Validation

Question	Answer
Where in Infrastructure Automation can I set default quota thresholds for users and groups?	These values can be set in the [`VM
Provisioning / StateMachines / ProvisionRequestQuotaVerification`] class instance of either the **Cloud** or **Infrastructure** namespace in your domain.	
Where in Infrastructure Automation can I set individual and group quota thresholds?	Tags can be assigned to groups or users by navigating to **Configuration > Access Control**. The following are valid tags that can be assigned to group or individual users: [**quota_max_cpu**, **quota_max_memory**, **quota_max_storage**].
Where can I customize the way our virtual machines are named?	Virtual machine naming conventions can be altered using the methods in the [`VM /
Provisioning / Naming`] class of either the **Cloud** or **Infrastructure** namespace in your domain.	

Question	Answer
How can I customize the email that is sent when a request is denied?	The Request Denied email message can be modified in the [`VM /
Provisioning / Email / MiqProvisionRequest_Denied`] in either the **Cloud** or **Infrastructure** namespace of your domain.	

Phase 4: Provisioning

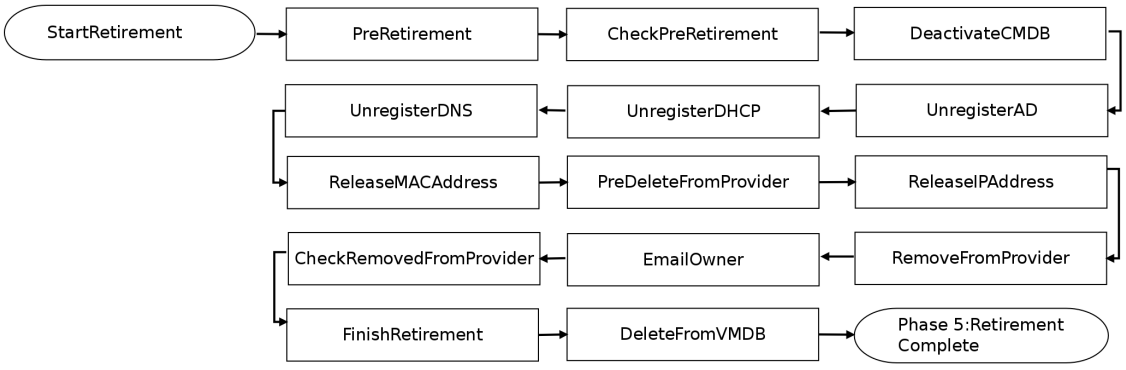
Target Type: Cloning a Template to a Virtual Machine.



Target Type: Clone to Template.

Question	Answer
Where can I customize the email message that is sent upon provisioning completion?	This can be customized using the [`VM / Provisioning / Email
/ MiqProvision_Complete`] in either the **Cloud** or **Infrastructure** namespace of your domain.	
Where can I change what is put into the virtual machines Annotation after provisioning?	These settings can be modified by leveraging the **`_PreProvision Ruby`** methods in [`VM / Provisioning / StateMachines /
Methods`] in either the *Cloud</emphasis> or **Infrastructure** namespace of your domain.	
Where do I set the default VC folder location for provisioning virtual machines?	This can be modified by leveraging by leveraging the **`_PreProvision Ruby`** methods in [`VM / Provisioning / StateMachines /
Methods`] in either the *Cloud</emphasis> or **Infrastructure** namespace of your domain.	
Where can I modify the virtual machine customization spec mapping?	This can be modified by leveraging by leveraging the `_PreProvision
Ruby **`methods in \[`**VM / Provisioning / StateMachines / Methods`] in either the *Cloud</emphasis> or **Infrastructure** namespace of your domain.	
Where can I modify the Clone_to_Template state_machine?	Navigate to [**`VM / Provisioning / StateMachines / VMProvision_VM / template`**] in either the **Cloud** or **Infrastructure** namespace of your domain.
Where can I modify the Clone_to_VM state_machine?	Navigate to [`VM /
Provisioning / StateMachines / VMProvision_VM / clone_to_vm`] in either the **Cloud** or **Infrastructure** namespace of your domain.	

Phase 5: Retirement



Question	Answer
Where can I customize the email message that is sent upon completion of virtual machine retirement?	This can be customized using the [`VM /
Retirement / Email / vm_retirement_emails`] in either the **Cloud** or **Infrastructure** namespace of your domain.	

Question	Answer
Where can I customize the email message that is sent during virtual machine retirement warnings?	This can be customized using the [`VM /
Retirement / Email / vm_retirement_emails`] in either the **Cloud** or **Infrastructure** namespace of your domain.	
If I want to customize what gets called during the retirement phase where should I look?	This can be customized using the [`VM / Retirement
/ StateMachines / VMRetirement`] in either the **Cloud** or **Infrastructure** namespace of your domain.	
How can I extend the virtual machine retirement date an additional number of days?	Create a custom button for virtual machines that launches [`/System/Request/vm_retire_extend`]. Then navigate to the [`VM /
Retirement / Email / vm_retire_extend`] Ruby method in the **Cloud** or **Infrastructure** namespaces and set the vm_retire_extend_days` value.	

Inline Method to Create a Provision Request

Ruby method

```
# Infrastructure Automation Automate Method
#
$evm.log("info", "Infrastructure Automation Automate Method  Building VM Provisioning Request Started")
#

prov= $evm.root['miq_provision']

  # arg1 = version
  args = ['1.1']
  # arg2 = templateFields
  args << {'name' => 'App'}
  # arg3 = vmFields
  args << {'vm_name' => 'CRM_APP', 'request_type' => 'template'}
  # arg4 =  requester
  args << {'owner_email' => 'admin@asd.com', 'owner_last_name' => 'Admin', 'owner_first_name' => 'Admin',
'user_name' => 'admin'}
  # arg5 =  tags
  args << {'crm' => 'true'}
  # arg6 =  WS Values
  args << nil
  # arg7 = emsCustomAttributes
  args << nil
  # arg8 = miqCustomAttributes
  args << nil

$evm.log("info","Building provisioning request with the following arguments: <#{args.inspect}>")
result = $evm.execute('create_provision_request', *args)
```

Migrating Custom Buttons

Migrating Custom Buttons

This workflow demonstrates how to export custom buttons from one Infrastructure Automation appliance and import them in another Infrastructure Automation appliance.

1. Export custom buttons from the source Infrastructure Automation appliance:
    1. SSH into the Infrastructure Automation appliance as `root`.
    2. Create a temporary directory:

       ```
       # mkdir /tmp/custom_buttons
       ```

    3. Browse to the Virtual Management Database (VMDB) directory:

       ```
       # vmdb
       ```

    4. Export custom buttons using the following `rake` command:

```
# rake evm:export:custom_buttons -- --directory /tmp/custom_buttons
```

5. Confirm the **yaml** file was created by navigating to the new directory:

```
# cd /tmp/custom_buttons
```

2. Copy the directory to the target Infrastructure Automation appliance:

```
scp -r /tmp/custom_buttons/ hostname:/tmp/custom_buttons
```

3. Import custom buttons on the target Infrastructure Automation appliance:

1. SSH to the target Infrastructure Automation appliance as **root**.
2. Browse to the VMDB directory:

```
# vmdb
```

3. Import custom buttons using the following **rake** command:

```
# rake evm:import:custom_buttons -- --source /tmp/custom_buttons
```

To verify successful import of custom buttons:

1. Log in to the target Infrastructure Automation appliance.
2. Browse to menu: **Automation > Automate > Customization**.
3. Click **Buttons** in the accordion menu.

Imported buttons will appear under the parent **Object Type**.

- General Configuration
  - Settings Overview
  - My Settings
    - Visual Settings
      - Grid and Tile Icons
      - Setting Default Items Per Page
      - Setting the Start Page
      - Setting Display Settings
    - Default Views
      - Setting Default Views for the User Interface
      - Setting Default Views for Services
      - Setting Default Views for Clouds
      - Setting Default Views for Infrastructure Components
      - Setting Default Views for Containers
    - Default Filters
      - Setting Default Filters for Cloud
      - Setting Default Filters for Containers
      - Setting Default Filters for Infrastructure
      - Setting Default Filters for Physical Infrastructure
      - Setting Default Filters for Services
    - Time Profiles
      - Creating a Time Profile
      - Editing a Time Profile
      - Copying a Time Profile
      - Deleting a Time Profile
  - Tasks
    - My VM and Container Analysis Tasks
      - Filtering the VM and Container Analysis Task List
      - Deleting a Task from the VM and Container Analysis Task List
    - My Other UI Tasks
      - Filtering the My Other UI Tasks List
      - Deleting a My Task
    - All Tasks
      - Filtering the VM and Container Analysis Task List
      - Deleting a VM and Container Analysis Task
    - All Tasks
      - Filtering the All Tasks List
      - Deleting a Task from the All Tasks List

- Configuration
  - Settings
    - Regions
      - Region Scope
      - Region Settings
      - Capacity and Utilization Collection
        - Capacity and Utilization Collection Settings
        - Enabling a Cluster, Host, or Datastore for Capacity and
      - Tags
        - Company Tag Categories and Tags
        - Creating a Tag Category
        - Deleting a Tag Category
        - Creating a Company Tag
        - Deleting a Company Tag
        - Importing Tags for Virtual Machines
        - Importing Tags for a Virtual Machine from a CSV File
        - Importing Custom Values for Virtual Machines and Hosts
        - Importing Asset Tags for a Virtual Machine from a CSV File
      - Tag Mapping
      - Customizing the Help Menu
    - Profiles
      - Creating an Analysis Profile
      - Creating a Host Analysis Profile
      - Creating a Virtual Machine Analysis Profile
      - Editing an Analysis Profile
      - Copying an Analysis Profile
      - Setting a Default Analysis Profile
    - Zones
      - Creating a Zone
      - Deleting a Zone
      - Editing a Zone
      - Adding SmartProxy Affinity to a Zone:
    - Servers
      - Changing Server Settings
        - Basic Information Settings
        - Server Control Settings
        - Server Roles
        - VMware Console Settings
        - NTP Servers Settings In the NTP Servers area, you can
        - Configuring SNMP
        - Outgoing SMTP Email Settings
        - Web Services Settings
        - Logging Settings
        - Custom Support URL Settings
      - Authentication
        - Changing Authentication Settings
      - Workers
        - Changing Settings for a Worker
      - Database
        - Changing a Database Setting
      - Customization and Logos
        - Custom Logos
        - Uploading a Custom Logo to the User Interface
        - Customizing the Login Background
        - Customizing the Login Panel Text
        - Displaying the Custom Configuration Settings
      - Advanced Settings
        - Editing Configuration Files Manually
        - Configuration Parameters
    - Schedules
      - Scheduling SmartState Analyses
        - Scheduling a SmartState Analysis or Compliance Check
        - Modifying a Schedule

        - Importing and Exporting Schedules
- Access Control
    - Creating a Tenant
    - Creating a Project
    - Managing Tenant and Project Quotas
    - Tagging Tenants and Projects
    - Creating a User
    - Deleting a User
    - Groups
    - Creating a Group
    - Roles
        - Account Roles and Descriptions
    - Creating a Role
- Diagnostics
    - Region Diagnostics
        - Server Role Priorities
    - Region Aware Server Roles
    - Setting the Priority of a Failover Role
    - Zone Diagnostics
        - Viewing the Status of Server Roles
        - Zone Aware Server Roles
            - Removing an Inactive Server
        - Zone Log Collections
            - Setting the Location of the Log Depot
            - Collecting and Downloading Logs from All Servers in a Zone
        - Capacity and Utilization Repair
            - Repairing Capacity and Utilization Data
    - Server Diagnostics
        - Workers
            - Reloading Worker Display
            - Restarting a Worker
        - Server and Audit Logs
            - Collecting Server Logs and Configuration Files
            - Setting the Location of the Log Depot for a Specific Server
            - Collecting the Current Log Set of a Server
            - Collecting All Log Sets from a Server
            - Viewing the Server, Audit, and Production Logs
            - Viewing the Server Log
            - Reloading the Server Log
            - Downloading the Server Log
            - Viewing the Audit Log
            - Reloading the Audit Log
            - Downloading the Audit Log
            - Viewing the Production Log
            - Reloading the Production Log
            - Downloading the Production Log
- Database Operations
    - Viewing Information on the VMDB
    - Database Regions and Replication
        - Creating a Region
    - Configuring Database Replication and Centralized Administration
        - Configuring a Remote Copy
        - Configuring the Global Copy
        - Resetting Database Replication
    - Backing Up and Restoring Infrastructure Automation in a Kubernetes Based Environment (podified)
    - Backing Up and Restoring a Database for an Appliance Based Environment
        - Running a Single Database Backup
        - Restoring a Database from a Backup
    - Performing a Binary Backup and Restoring the Database
        - Performing a Binary Backup
        - Restoring a Database from the Binary Backup
    - Running Database Garbage Collection
    - Changing the Database Password

        - Changing the Password on the Database Appliance
        - Changing the Password on the Worker Appliances
      - Adding a New Appliance to an Existing Region with a Non-default Password
      - Creating a Database Dump
      - Enabling Events Syndication
    - SmartProxies
      - Installing the SmartProxy from the Console
      - Entering Credentials and Operating System for the Target Host
    - About
- Default Roles
- EvmRole-super_administrator
- EvmRole-administrator
- EvmRole-approver
- EvmRole-auditor
- EvmRole-desktop
- EvmRole-operator
- EvmRole-security
- EvmRole-support
- EvmRole-user
- EvmRole-user_limited_self_service
- EvmRole-user_self_service
- EvmRole-vm_user
- EvmRole-tenant_administrator
- EvmRole-tenant_quota_administrator
- EvmRole-consumption_administrator
- EvmRole-container_administrator
- EvmRole-container_operator
- EvmRole-reader

General Configuration

Settings Overview

Use the **Settings** menus of the Infrastructure Automation user interface to view or modify the global configuration of your appliance.

Settings

To view or modify global settings for your appliance, click **Settings > My Settings** to modify. The availability of each menu item depends on the role assigned to your user account. For more information on roles, see Roles.

The following menu items are available from the settings menu:

- **My Settings**

  This menu is available to all Infrastructure Automation users. The settings in this menu control how elements in the user interface are displayed, time profiles, and tags for the currently logged-in individual user.

- **Tasks**

  This menu allows you to view virtual machine SmartState Analysis tasks that can be tracked through the console. The status of each task is displayed, including time started, time ended, what part of the task is currently running, and any errors encountered.

Application Settings

The **Application Settings** menu is now accessible from **Settings > Application Settings** from the user interface.

This menu allows you to specify enterprise, region, zone, and server settings for your Infrastructure Automation infrastructure. Diagnostics such as logs and the status of processes are also shown here.

My Settings

The options under the **My Settings** menu allow you to configure user interface display options specific to the user account with which you log in to Infrastructure Automation, such as the default view displayed on login, and personal tags. You can also configure the color scheme, and button options on the main dashboard.

These options are available by clicking **Settings > My Settings**.

Visual Settings

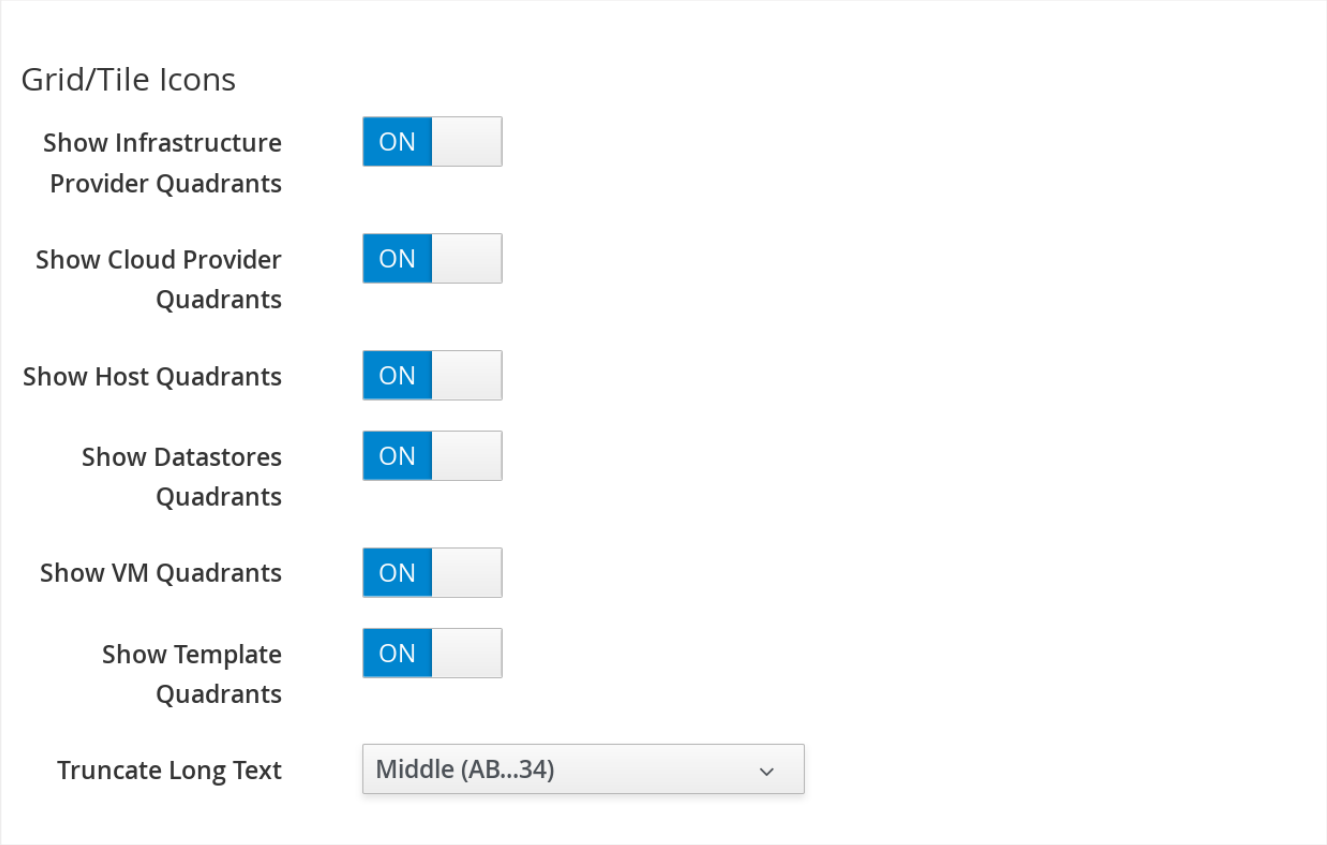
The options under the **Visual** menu allow you to configure how user interface elements are displayed in the web user interface. For all options, click **Save** to update the settings, or click **Reset** to undo any unsaved changes that have been made on the current screen.

Grid and Tile Icons

This group of settings is used to control the view of your virtual thumbnails. Each thumbnail can be viewed as a single icon or as an icon with four quadrants. Use the quadrant view to see a component's properties at a glance.

Use the following procedure to change grid and tile icons:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Visual** tab.
2. In **Grid/Tile Icons**, set items to **ON** to display all four quadrants for the item, or **OFF** to display only one icon.
3. Click **Save**.



- Set **Show Infrastructure Provider Quadrants** to **ON** to see the four icons in your provider under menu: **Compute > Infrastructure > Providers**.
- Set **Show Cloud Provider Quadrants** to **ON** to see the four icons in your hosts under menu: **Cloud > Providers**.
- Set **Show Host Quadrants** to **ON** to see the four icons in your hosts under menu: **Compute > Infrastructure > Hosts**.
- Set **Show Datastores Quadrants** to **ON** to see the four icons in your datastores under menu: **Compute > Infrastructure > Datastores**.
- Set **Show VM Quadrants** to **ON** to see the four icons in your virtual machines under menu: **Compute > Infrastructure > Virtual Machines**.
- Set **Show Template Quadrants** to **ON** to see the four icons in your templates under menu: **Compute > Infrastructure > Virtual Machines > Templates**.

Set any of the above options to **OFF** to see only one icon instead of four quadrants.

- Use the **Truncate Long Text** list to specify how the names of items are displayed if they are too long to show in full. Select the option based on the pattern shown.

Setting Default Items Per Page

Use the following procedure to set the default number of items to display on each resource page.

1. From the **Settings** menu, browse to **My Settings**, then click on the **Visual** tab.
2. In the **Default Items Per Page** area, select the default number of items to display for each view from the corresponding drop down list.

3. Click **Save**.

Setting the Start Page

Use the following procedure to set the default start page after logging in. For example, instead of going to the **Infrastructure Automation** dashboard, you can set the default start page to see a list of your virtual machines.

1. From the **Settings** menu, browse to **My Settings**, then click on the **Visual** tab.
2. In the **Start Page** area, select the page to display at login.
3. Click **Save**.

Setting Display Settings

Use the following procedure to set your own themes, colors, and time zone for the console. These settings are specific to the logged-on user.

1. From the **Settings** menu, browse to **My Settings**, then click on the **Visual** tab.
2. Make selections from **Display Settings** for the following items:
    1. Use **Chart Theme** to select a group of colors and font sizes specifically for charts.
    2. Use **Time Zone** to select the time zone in which to display the console.

       Note:

       In time zones where clocks are set forward for daylight savings time, the time zone correctly displays as EDT (Eastern Daylight Time) in the console. When the clocks are set back, it correctly displays as EST (Eastern Standard Time).
    3. Use **Locale** to select the language in which to display the console.
3. Click **Save**.

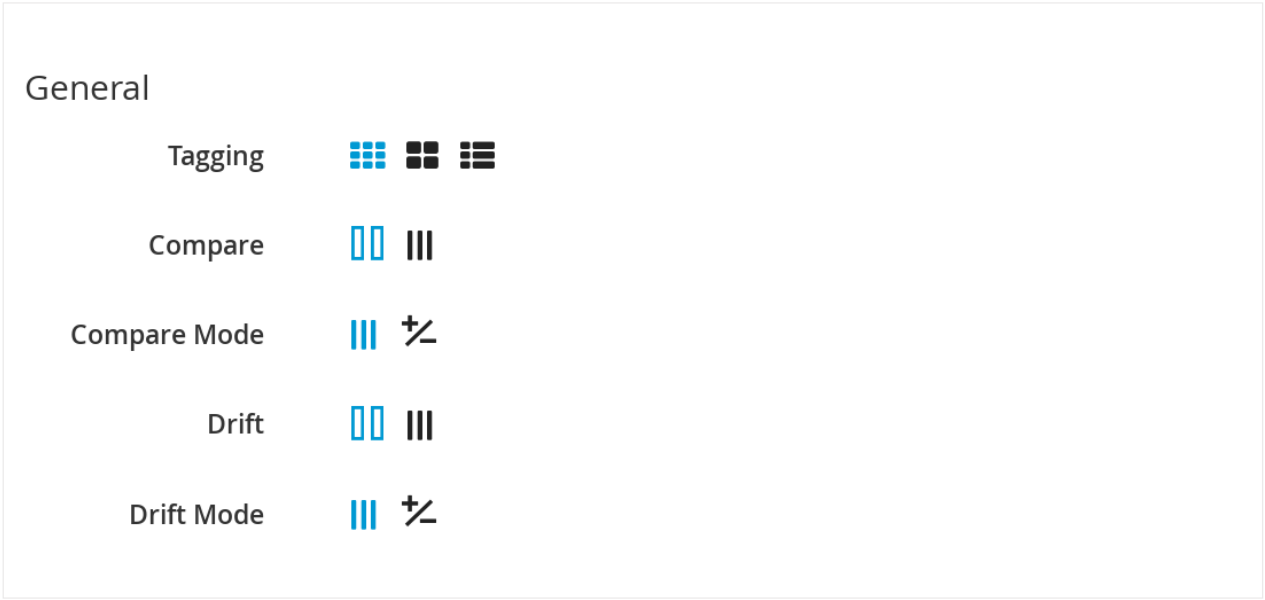
Default Views

The options under the **Default Views** menu allow you to configure the default layout used to display individual screens in the Infrastructure Automation user interface. The options you select under this menu specify the default options for each screen, but you can also change the layout for each screen using the layout buttons on each screen.

Setting Default Views for the User Interface

Use the following procedure to set general view options:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Views** tab.
2. In the **General** area, click the appropriate button for the way you want to view each type of screen listed. The selected view shows as a blue icon.



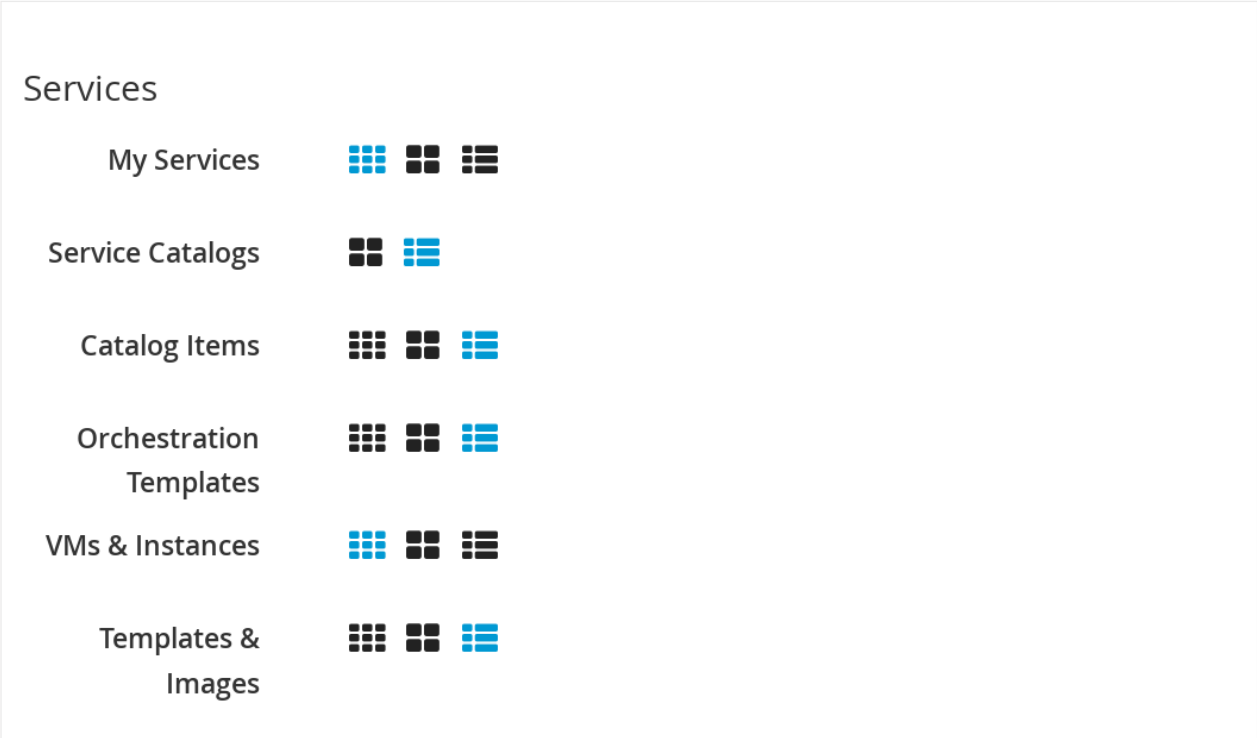
3. Click **Grid View** to view virtual thumbnails or icons.
4. Click **Tile View** for a view that combines the virtual thumbnail with some text properties that describe the items.
5. Click **List View** to view a detailed text listing.
6. Click **Expanded View** for an expanded view.
7. Click **Compressed View** for a compressed view.
8. Click **Exists Mode** to view only whether an attribute exists or not.
9. Click **Save**.

Setting Default Views for Services

Use the following procedure to set default views for services in the menu: **Compute > Services** tab.

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Views** tab.

2. In the **Services** area, click the appropriate button for the way you want to view each item.



- Click **Grid View** to view virtual thumbnails or icons.
- Click **Tile View** for a view that combines the virtual thumbnail with some text properties that describe the items.
- Click **List View** to view a text listing.

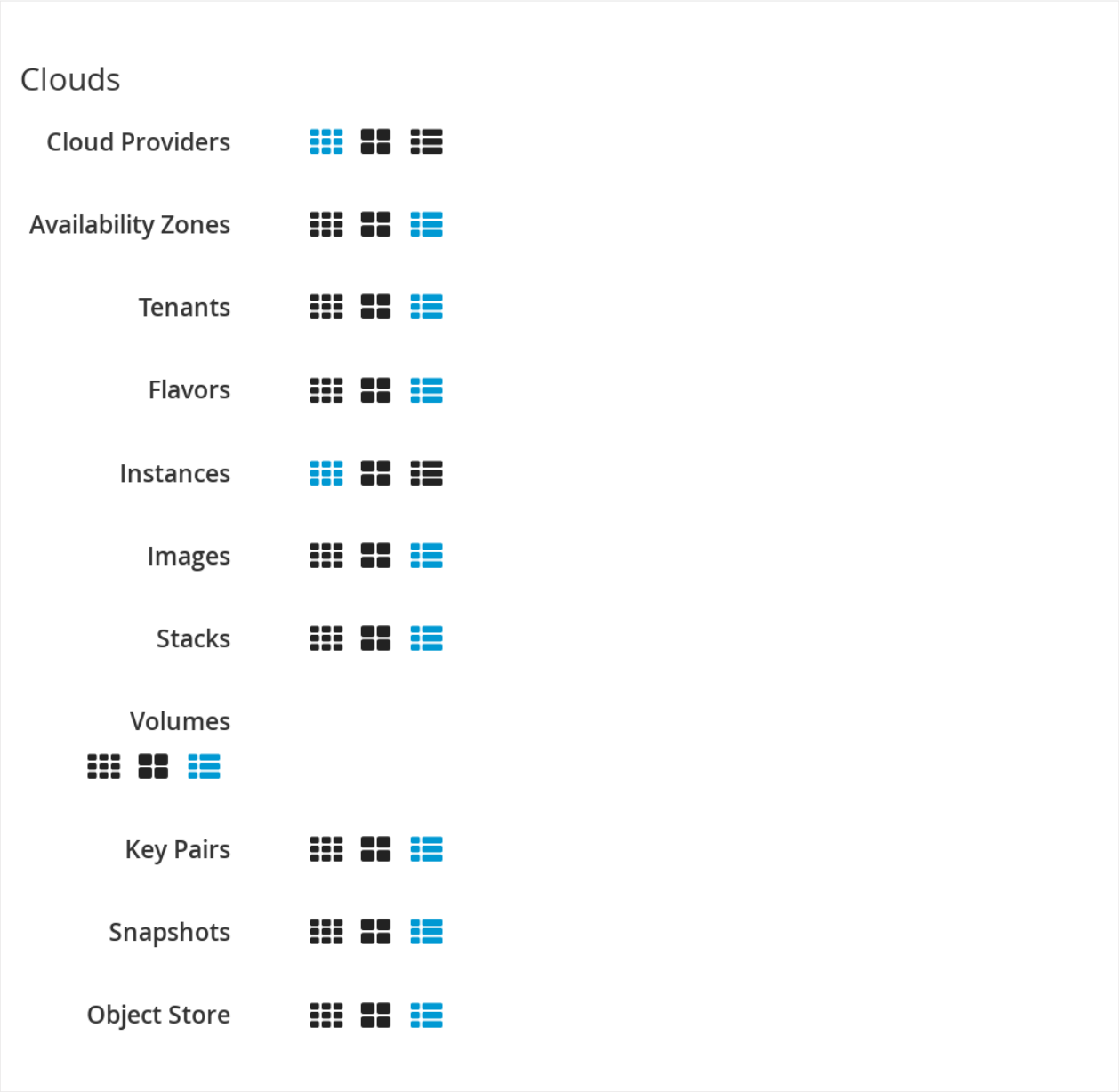
3. Click **Save**.

Setting Default Views for Clouds

Use the following procedure to set default views for clouds in the menu: **Compute > Clouds** tab.

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Views** tab.

2. In the **Clouds** area, click the appropriate button for the way you want to view each item.



- Click **Grid View** to view virtual thumbnails or icons.
- Click **Tile View** for a view that combines the virtual thumbnail with some text properties that describe the items.
- Click **List View** to view a detailed text listing.

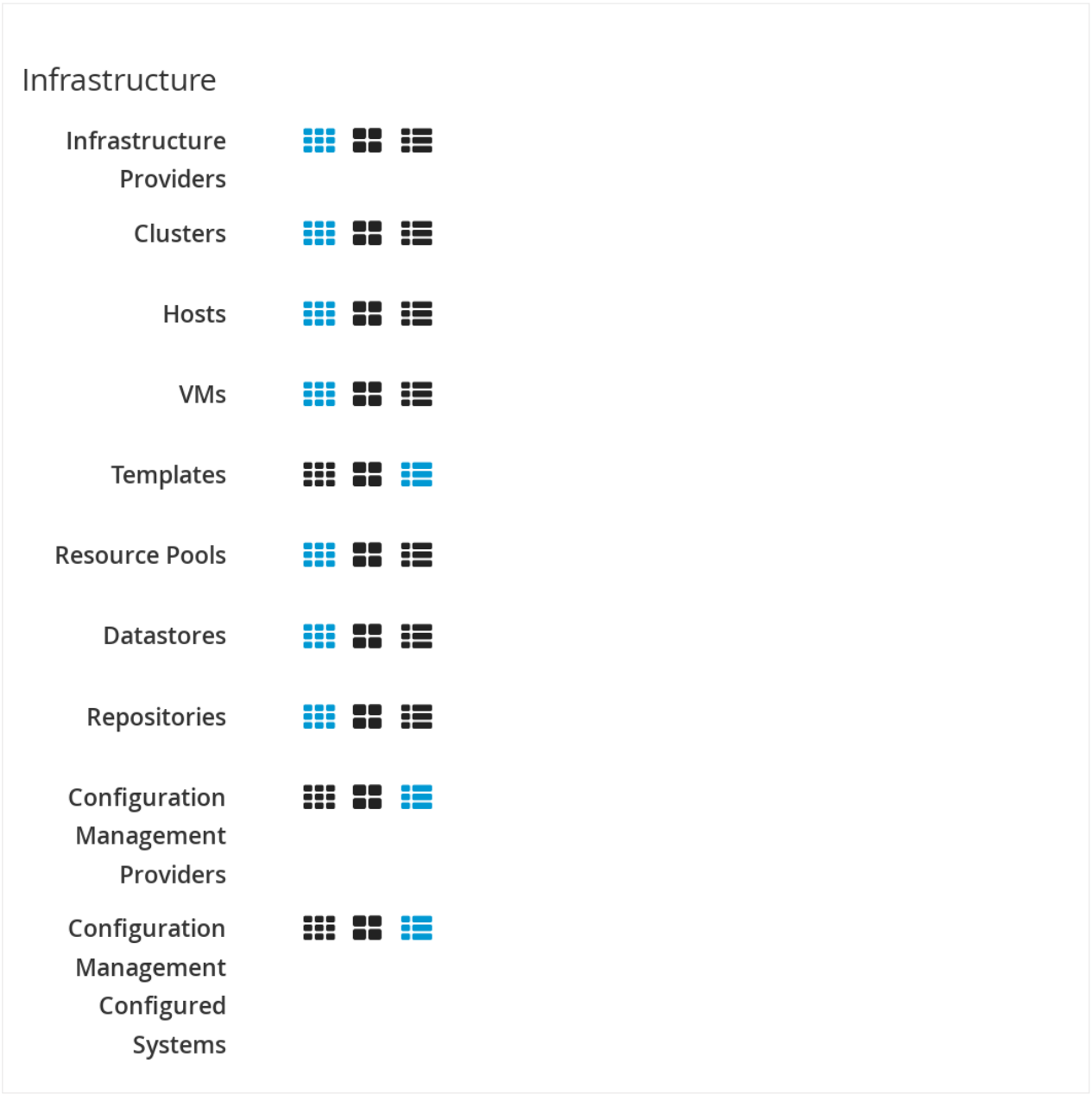
3. Click **Save**.

Setting Default Views for Infrastructure Components

Use the following procedure to set default views for infrastructure components in the menu: **Compute > Infrastructure** tab.

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Views** tab.

2. In the **Infrastructure** area, click the appropriate button for the way you want to view each item.



- Click **Grid View** to view virtual thumbnails or icons.
- Click **Tile View** for a view that combines the virtual thumbnail with some text properties that describe the items.
- Click **List View** to view a detailed text listing.

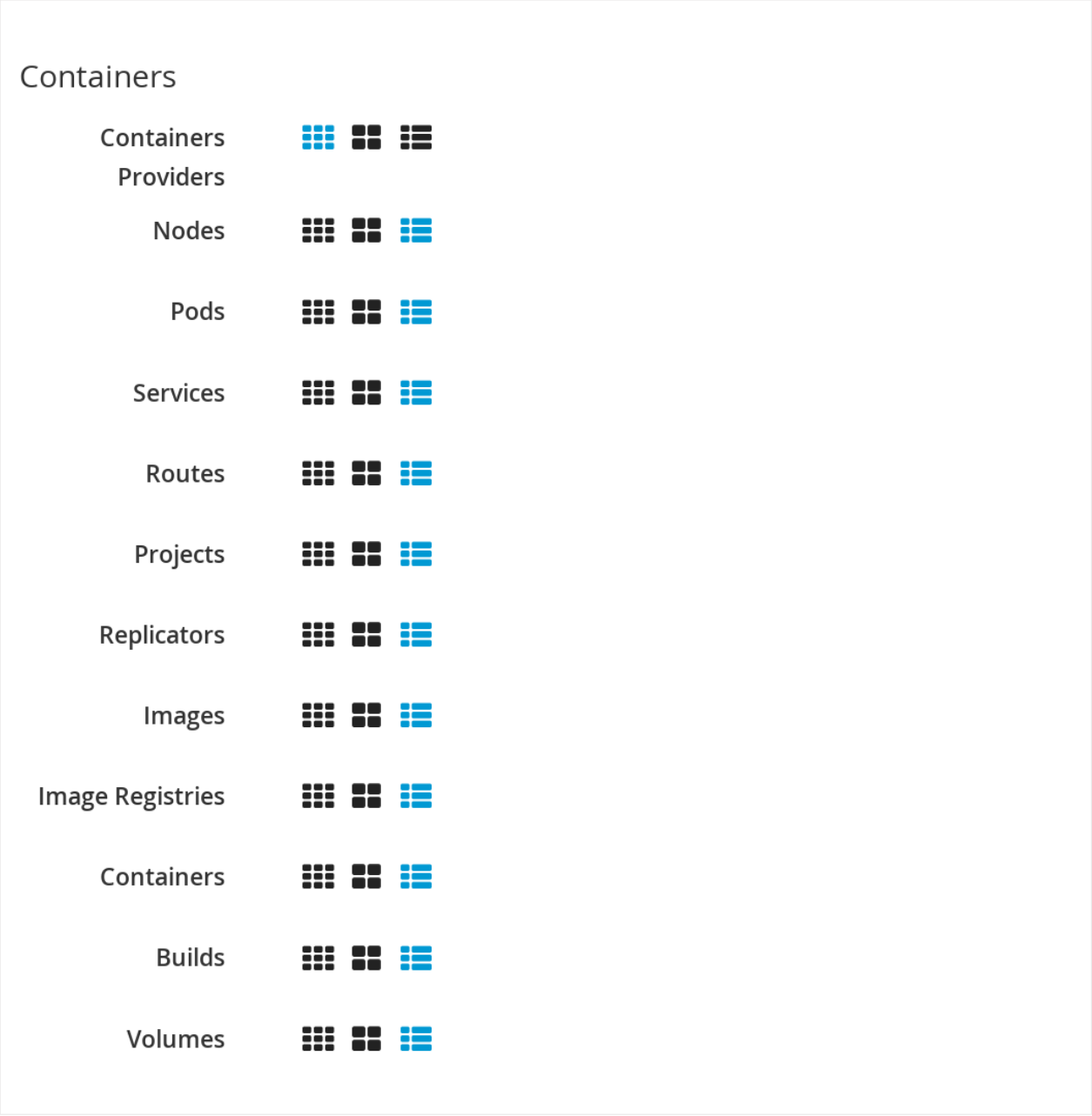
3. Click **Save**.

Setting Default Views for Containers

Use the following procedure to set default views for containers in the menu: **Compute > Containers** tab.

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Views** tab.

2. In the **Containers** area, click the appropriate button for the way you want to view each item.



- Click **Grid View** to view virtual thumbnails or icons.
- Click **Tile View** for a view that combines the virtual thumbnail with some text properties that describe the items.
- Click **List View** to view a text listing.

3. Click **Save**.

Default Filters

The options in the **Default Filters** menu allow you to configure the default filters displayed for your hosts, virtual machines, and templates. These settings are available to all users.

Setting Default Filters for Cloud

To set default filters for cloud components:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Filters** tab.
2. From the **Cloud** folder, check the boxes for the default filters that you want available. Items that have changed show in blue text.

3. Click **Save**.

Setting Default Filters for Containers

To set default filters for containers:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Filters** tab.
2. From the **Containers** folder, check the boxes for the default filters that you want available. Items that have changed show in blue text.
3. Click **Save**.

Setting Default Filters for Infrastructure

To set default filters for infrastructure components:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Filters** tab.
2. In the **Infrastructure** folder, select the default filters that you want available. Items that have changed show in blue text.
3. Click **Save**.

Setting Default Filters for Physical Infrastructure

To set default filters for physical infrastructure components:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Filters** tab.
2. In the **Physical Infrastructure** folder, select the default filters that you want available. Items that have changed show in blue text.
3. Click **Save**.

Setting Default Filters for Services

To set default filters for services:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Default Filters** tab.
2. In the **Services** folder, select the default filters that you want available. Items that have changed show in blue text.
3. Click **Save**.

Time Profiles

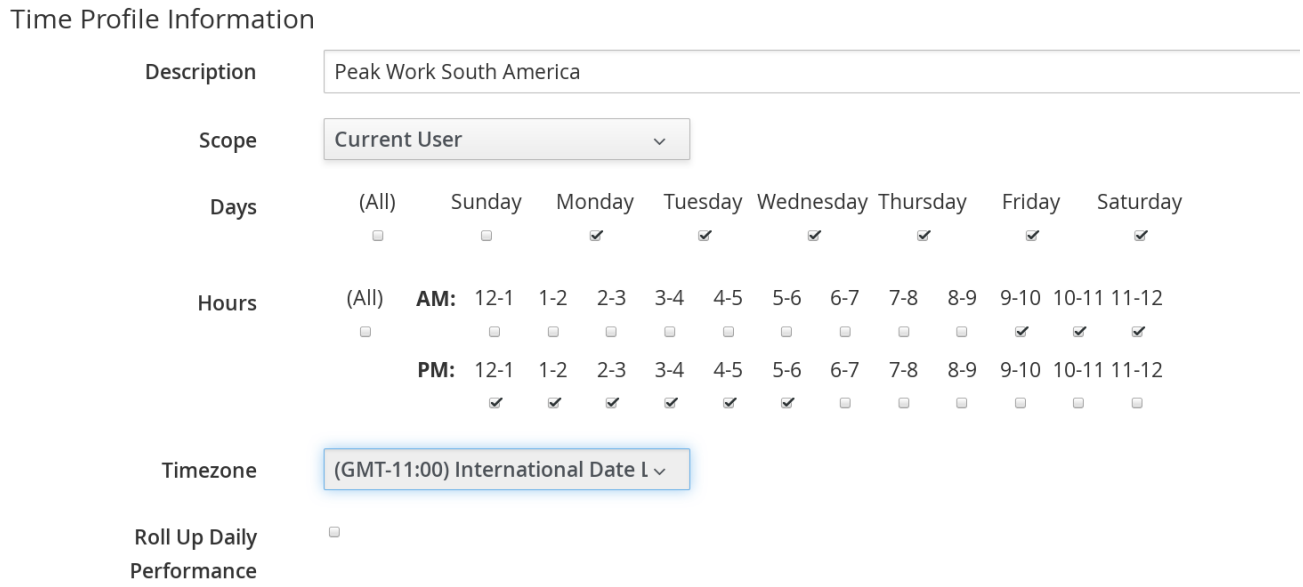
The options under the **Time Profiles** menu allow you to specify the hours for which data is displayed when viewing capacity and utilization screens. Time profiles are also used to configure performance and trend reports.

Creating a Time Profile

To create a time profile:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Time Profiles** tab.

2. Click **Configuration**, and **Add a new Time Profile**.



3. Type a meaningful name in the **Description** field.
4. Select the users who can access the time profile from the **Scope** list:
   - Select **All Users** to create a time profile that is available to all users. Only the super administration and administration roles can create, edit, and delete a global profile.
   - Select **Current User** if this time profile should only be available to the user creating it.
5. Check the **Days** and **Hours** for the time profile.
6. For **Timezone**, you can select a specific time zone or, you can let the user select a time zone when displaying data.
7. If you select a specific time zone, you also have the option to **Roll Up Daily Performance** data. This option is only available to users with the administration or super administration role. Enabling the **Roll Up Daily Performance option** reduces the time required to process daily capacity and utilization reports and to display daily capacity and utilization charts.
8. Click **Add**.

Note:

The following relationships exist between time zones and performance reports:

- The configured time zone in a performance report is used to select rolled up performance data, regardless of the user's selected time zone.
- If the configured time zone is null, it defaults to UTC time for performance reports.
- If there is no time profile with the report's configured time zone that is also set to roll up capacity and utilization data, the report does not find any records.

For non-performance reports, the user's time zone is used when displaying dates and times in report rows.

Editing a Time Profile

To edit a time profile:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Time Profiles** tab.
2. Check the time profile you want to edit.
3. Click **Configuration**, and **Edit selected Time Profile**.
4. Make the required changes.
5. Click **Save**.

Copying a Time Profile

To copy a time profile:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Time Profiles** tab.
2. Check the time profile you want to copy.
3. Click **Configuration**, and **Copy selected Time Profile**.
4. Make the required changes.
5. Click **Save**.

Deleting a Time Profile

To delete a time profile:

1. From the **Settings** menu, browse to **My Settings**, then click on the **Time Profiles** tab.
2. Check the time profile you want to delete.
3. Click **Configuration**, and **Delete selected Time Profiles**.
4. Click **Save**.

Tasks

The options under the **Tasks** menu allow you to view and control currently running tasks in Infrastructure Automation. The status of each task is displayed, including time started, time ended, what part of the task is currently running, and any errors encountered.

My VM and Container Analysis Tasks

The **My VM and Container Analysis Tasks** menu allows you to view all tasks running on virtual machines and containers for the currently logged-in user.

From the **My VM and Container Analysis Tasks** menu, you can:

- See jobs that the logged on user created for the SmartProxy either through a schedule or by manually initiating a SmartState Analysis of a virtual machine or container.
- See if a job completed successfully, resulted in an error, or is running.
- See the reason for an error.
- Filter the tasks by status and state.
- View the owner or host of the virtual machine or container referenced.
- Delete a task either explicitly or older than another task.

Filtering the VM and Container Analysis Task List

This procedure describes how to filter the virtual machine and container analysis task list. You can filter the task list by zone, time period, task status, and task state.

To filter the virtual machine and container analysis task list:

1. From the **Settings** menu, select **Tasks**.
2. Click **My VM and Container Analysis Tasks**.
3. From the **Zone** list, select a specific zone, or select **All Zones**.
4. From the **24 Hour Time Period** list, select the period of time to view the tasks.
5. For **Task Status**, select the check boxes next to the status to view.
6. From the **Tasks State** list, select the state to view.
7. Click **Apply**.

Deleting a Task from the VM and Container Analysis Task List

To delete a task from the list of virtual machine and container analysis tasks:

1. From the **Settings** menu, select **Tasks**.
2. Click **My Tasks**.
3. Select the tasks to delete. From the **Delete** menu:
    1. Click **Delete** to delete the selected tasks.
    2. Click **Delete Older** to delete the tasks older than the selected tasks.
    3. Click **Delete All** to delete all tasks.
4. Click **OK**.

My Other UI Tasks

The **My Other UI Tasks** menu allows you to view all tasks running in the user interface for the currently logged-in user.

Filtering the My Other UI Tasks List

This procedure describes how to filter the user interface analysis task list. You can filter the task list by time period, task status, and task state.

To filter the other user interface task list:

1. From the settings menu, select **Tasks**.
2. Click **My Other UI Tasks**.
3. From the **24 Hour Time Period** list, select the period of time to view the tasks.
4. For **Task Status**, select the check boxes next to the status to view.
5. From the **Tasks State** list, select the state to view.
6. Click **Apply**.

Deleting a My Task

To delete a task from the list of other user interface tasks:

1. From the settings menu, select **Tasks**.
2. Click **My Tasks**.
3. Select the tasks to delete. From the **Delete** menu:
    1. Click **Delete** to delete the selected tasks.
    2. Click **Delete Older** to delete the tasks older than the selected tasks.
    3. Click **Delete All** to delete all tasks.
4. Click **OK**.

All Tasks

The **All Tasks** menu allows you to view all tasks running on virtual machines and containers for all users.

From the **All Tasks** menu, you can:

- See jobs that all users have created for the SmartProxy either through a schedule or by manually initiating a SmartState Analysis of a virtual machine or container.
- See if a job completed successfully, resulted in an error, or is running.
- See the reason for an error.
- Filter the tasks by status and state.
- View the owner or host of the virtual machine or container referenced.
- Delete a task either explicitly or older than another task.

Filtering the VM and Container Analysis Task List

This procedure describes how to filter virtual machine analysis task lists. You can filter the task list by zone, time period, task status, and task state.

To filter the virtual machine and container analysis task list:

1. From the **Settings** menu, select **Tasks**.
2. Click **All Tasks**.
3. From the **Zone** list, select a specific zone, or select **All Zones**.
4. From the **24 Hour Time Period** list, select the period of time to view the tasks.
5. For **Task Status**, select the check boxes next to the status to view.
6. From the **Tasks State** list, select the state to view.
7. Click **Apply**.

Deleting a VM and Container Analysis Task

To delete a task from the list of all virtual machine and container analysis tasks:

1. From the settings menu, select **Tasks**.
2. Click **All Tasks**.
3. Select the tasks to delete. From the **Delete** menu:
    1. Click **Delete** to delete the selected tasks.
    2. Click **Delete Older** to delete the tasks older than the selected tasks.
    3. Click **Delete All** to delete all tasks.
4. Click **OK**.

All Tasks

The **All Tasks** menu allows you to view all tasks for all users. This menu is only accessible when you are logged in as the **admin** user.

Filtering the All Tasks List

This procedure describes how to filter the list of all other tasks. You can filter the task list by time period, task status, and task state.

To filter the all other tasks list:

1. From the **Settings** menu, select **Tasks**.
2. Click **All Tasks**.
3. From the **24 Hour Time Period** list, select the period of time to view the tasks.
4. For **Task Status**, select the check boxes next to the status to view.
5. From the **Tasks State** list, select the state to view.
6. Click **Apply**.

Deleting a Task from the All Tasks List

To delete a task from the list of all other tasks:

1. From the settings menu, select **Tasks**.
2. Click **All Tasks**.
3. Select the tasks to delete. From the **Delete** menu:
    1. Click **Delete** to delete the selected tasks.
    2. Click **Delete Older** to delete the tasks older than the selected tasks.
    3. Click **Delete All** to delete all tasks.
4. Click **OK**.

Configuration

The options under the **Configuration** menu allow you to configure global options for your Infrastructure Automation environment, view diagnostic information, and view analytics on the servers in the environment. The menu displays the Infrastructure Automation environment at the enterprise, zone, and server levels.

There are four main areas:

- **Settings**

  This menu allows you to configure global settings for your Infrastructure Automation infrastructure. You can also create analysis profiles and schedules for these profiles.

- **Access Control**

  This menu contains options for configuring users, groups, roles, and tenants.

- **Diagnostics**

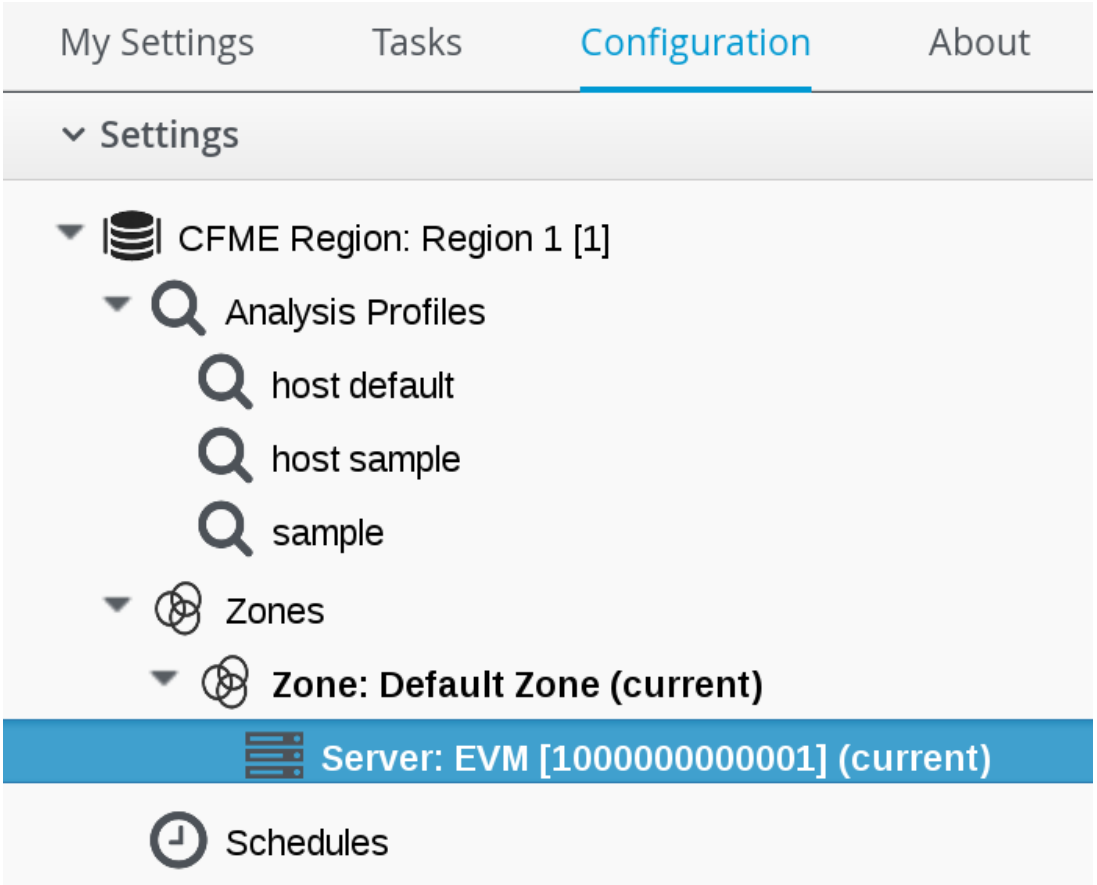
  This menu displays the status of your servers and their roles and provides access to logs.

- **Database**

  specify the location of your Virtual Machine Database (VMDB) and its login credentials.

Settings

The options under the **Settings** area provide a hierarchical view of options that allow you to configure global options for the infrastructure of your Infrastructure Automation environment. At the top level, you have **Settings** including users, LDAP Groups, account roles, capacity and utilization collection, tag categories, values, and imports, custom variable imports, and license uploads. When you click on **Settings** and expand it, you can configure **Analysis Profiles**, **Zones**, and **Schedules**.



When you go the **Settings** accordion, you are automatically taken to the server list under **Zones**.

Regions

Use **Regions** for centralizing data which is collected from public and private virtualization environments. A region is ultimately represented as a single database for the VMDB. Regions are particularly useful when multiple geographical locations need to be managed as they enable all the

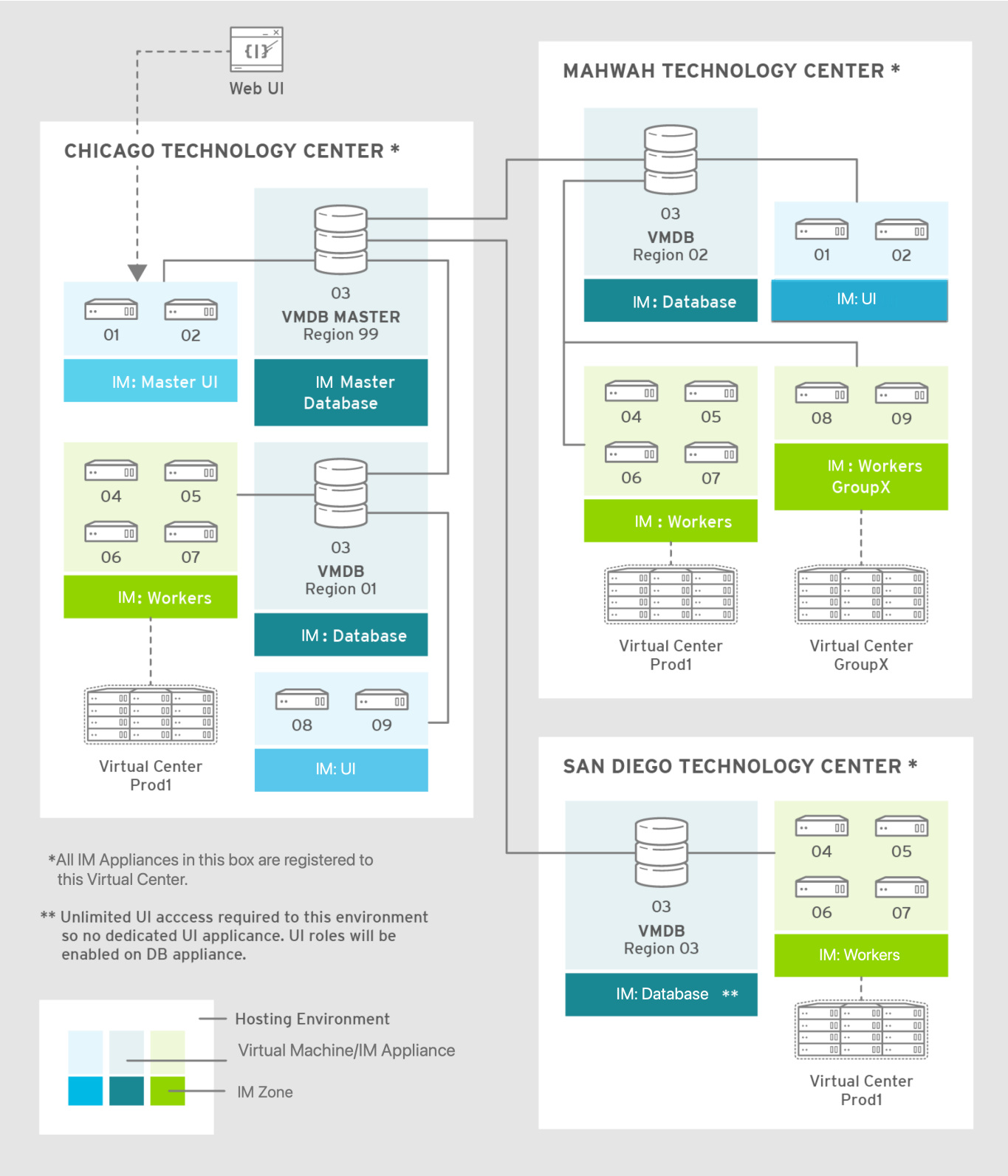
data collection to happen at each particular location and avoid data collection traffic across slow links between networks.

When multiple regions are being used, each with their own unique ID, a master region can be created to centralize the data of all the children regions into a single master database. To do this, configure each child region to replicate its data to the master region database (Red Hat recommends use of region 99). This parent and child region is a one-to-many relationship.

Regions can contain multiple zones, which in turn contain appliances. Zones are used for further segregating network traffic along with enabling failover configurations. Each appliance has the capability to be configured for a number of specialized server roles. These roles are limited to the zone containing the appliance they run on.

Only one failover type of each server role can run in a zone. If multiple appliances have the same failover role, the extras are used as backups that activate only if the primary appliance fails. Non-failover server roles can run on multiple appliances simultaneously in a zone, so resources can be adjusted according to the workload those roles are responsible for.

The following diagram demonstrates an example of the multiple regions working together in a Infrastructure Automation environment.

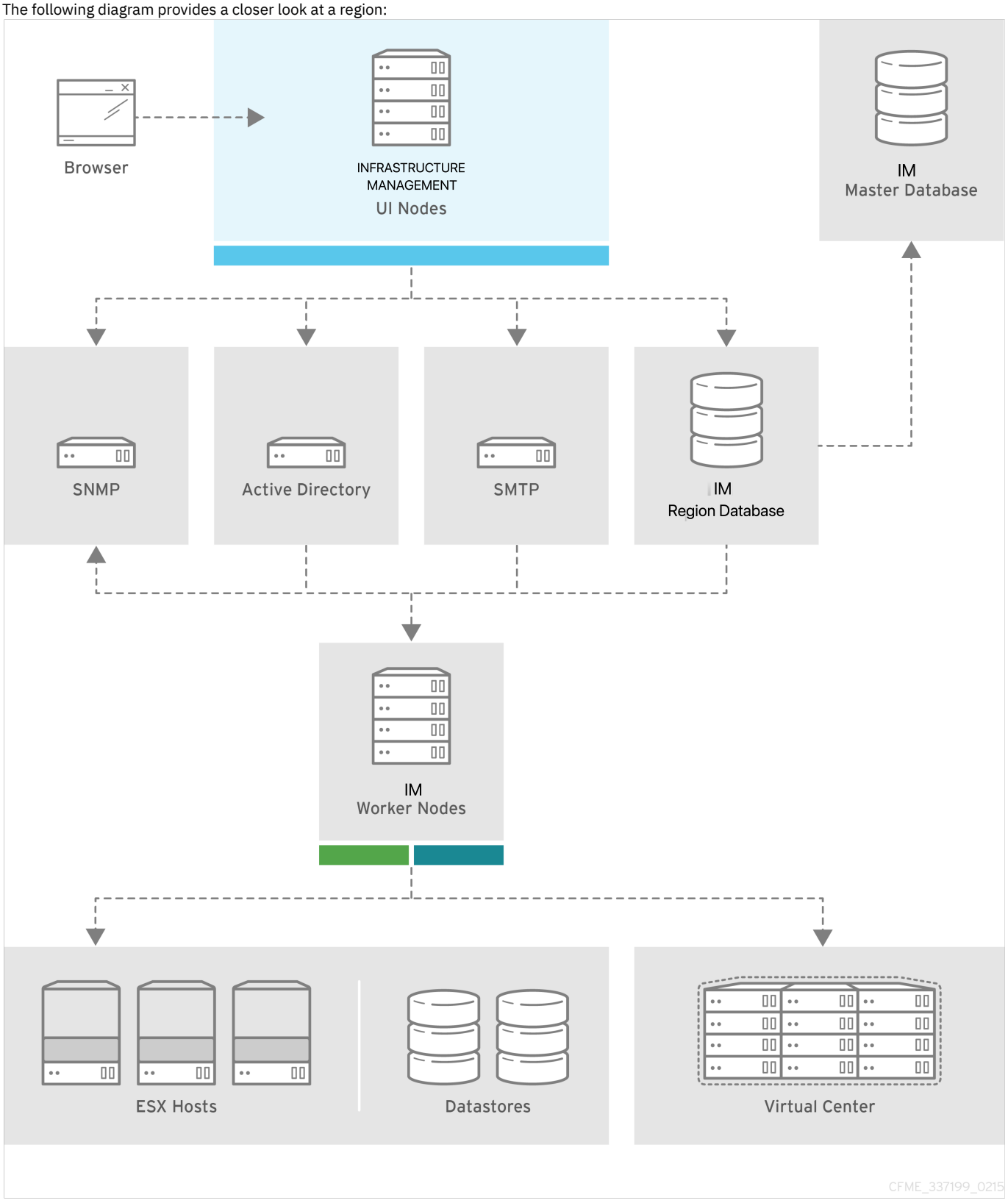


The Master appliance is located in Chicago and contains a master region and a subregion that manages the worker appliances. The Mahwah technology center contains a single subregion that manages two zones. Likewise the San Diego technology center contains a single subregion managing a single zone.

Note:

- Replicating a parent region to a higher-level parent is not supported.
- Parent regions can be configured after the child regions are online.

The following diagram provides a closer look at a region:



In this region, we have several Infrastructure Automation appliances acting as UI nodes and worker nodes. These worker nodes execute tasks on the providers in your environment. The Region also uses a region database that reports to a master database on the main Infrastructure Automation appliance. All appliances can connect to the authentication services (Active Directory, LDAP, Identity Management), outgoing mail (SMTP), and network services (SNMP).

Region Scope

Regions are used to consolidate data from multiple VMDBs to a central database. The database at the top level, the master VMDB, cannot be used for operational tasks such as SmartState Analysis or Capacity and Utilization data collection. It is intended for use as a reporting database that

includes all information across multiple subordinate regions. The subordinate regions replicate their information to the master.

Note:

The subordinate regions are not aware of each other from a database perspective. You cannot see information from one subordinate region in another. The only VMDB with data visibility to all subordinate regions is the top level.

Master Regions Scope

- Reports all information from all subordinate VMDBs reporting up to it.
- Can perform power operations on virtual machines from subordinate regions.
- Controls its own access control list.

Subordinate Regions Scope

- Each subordinate controls its own access control independent of the other regions.
- Can only do work (such as SmartState Analysis and Capacity and Utilization collection) in its own region.
- Has no knowledge of the other regions.
- Replicates its data up to the master region.

Region Settings

In the **Region** area, set items that apply to your entire Infrastructure Automation infrastructure such as users, LDAP Groups, capacity and utilization collection, company tags and tag categories, and licensing. Regions are also used for database replication.

Capacity and Utilization Collection

Capacity and Utilization Collection Settings

Use **C & U Collection Settings** to select specifically which clusters and datastores you want to collect usage data for. By selecting a cluster, you are choosing to collect data for all hosts and virtual machines that are part of that cluster. You must also have a server with the Capacity & Utilization **Coordinator**, **Data Collector**, and **Data Processor** roles enabled as well. See Section **Server Control Settings**.

After a provider has been discovered and its relationships refreshed, view the clusters, hosts, and datastores from the settings menu. Navigate to **Settings**, then click on **Application Settings**. Click the menu: **Settings > Region**, then select the **C & U Collection** tab.

Enabling a Cluster, Host, or Datastore for Capacity and

Utilization Collection

To enable a cluster, host, or datastore for Capacity and Utilization Collection:

1. Click **Settings**, then click on **Application Settings**.
2. Click the **Settings** accordion, then select **Region**, then click on the **C & U Collection** tab.
3. In the **Clusters** area, check all clusters and hosts that you want to collect data for.
4. In the **Datastores** area, check all datastores that you want to collect data for.
5. Click **Save**.

Note:

1. As new clusters, hosts, and datastores are discovered, you will need to come back to this configuration to enable collection of capacity and utilization data unless you have used the **Collect for All** check boxes.
2. **Collect for All Clusters** must be checked to be able to collect capacity and utilization data from cloud providers such as Red Hat OpenStack Platform or Amazon EC2.

Tags

Company Tag Categories and Tags

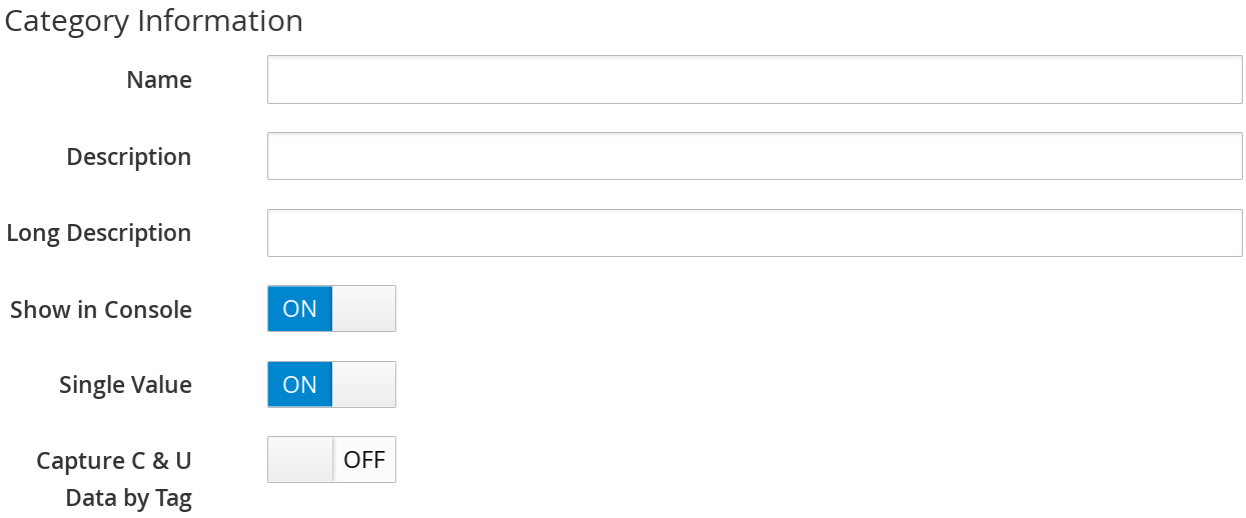
Infrastructure Automation allows you to create your own set of tags and tag categories. Use tags to create a customized, searchable index for your resources. Depending on your database type, your tags may be case sensitive. After creating these values, you can apply them to your resources. There are two kinds of tags.

- **Company tags** which you will see under **My Company Tags** for a resource. Create company tags from the settings menu. Browse to **Settings > Application Settings**. Click the **Settings** accordion, then select the **Region** and click on the **Tags** tab, then **My Company Tags**. A selection of company tags is provided to you by default as samples. These can be deleted if you do not need them, but are not recreated by Infrastructure Automation.
- **System tags** are assigned automatically by Infrastructure Automation.

Creating a Tag Category

To create a tag category:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then **Region**.
3. Select the **Tags** tab, then click on the **My Company Categories** tab.
4. Click **Add** under the **Actions** column to create a new category.
5. In the **Category Information** area:



   - Use **Name** to create a short name that refers to category in the VMDB.

     Note:

     The **Name** and **Single Value** fields cannot be changed after the category has been added.
   - Use **Description** to type a brief explanation of how the category should be used. This shows when you try to add a value to the category.
   - Use **Long Description** to type a detailed explanation of the category.
   - Set **Show in Console** to ON when the category is ready for use in the console. For example, you want to populate values for the category before exposing it to users.
   - Set **Single Value** to ON for categories that can only have a single value assigned to a resource. For example, a virtual machine can only be assigned to one location, but could belong to more than one department. This cannot be changed after the category is created.
   - Set **Capture C & U Data by Tag** to ON for the ability to group capacity and utilization data by this tag category. To use this, be sure to assign this tag to all the resources that you want to group by.
6. Click **Add**.

Repeat these steps for each category you need. After you have created the category, you can add values to it.

Important

If no values are created for a category, you are unable to assign a value from that category nor be able to filter by that category.

Deleting a Tag Category

To delete a tag category:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then **Region**.
3. Select the **Tags** tab, then click on the **My Company Categories** tab.
4. Click **Delete** under the **Actions** column for the category you want to delete.
5. Click **OK** to confirm.

Note:

When you delete a tag category, the category values are removed, and any tags from the category are unassigned from all resources.

Creating a Company Tag

To create a company tag:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then **Region**.
3. Select the **Tags** tab, then click on the **My Company Tags** tab.
4. In the **Choose a Category** area, select a category from the **Category** list.

   Note:

   - Some categories only allow one value to be assigned to a resource.
   - For some databases such as **PostgreSQL**, tags are case sensitive. For example, filtering by 'Linux' in title case give you different results from filtering by 'linux' in lower case.
5. Click **Add** under the **Actions** column, and type a **Name** and **Description** for your new value.
6. Click **Add** once again to add the new entry to the table.

Deleting a Company Tag

To delete a company tag:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then **Region**.
3. Select the **Tags** tab, then click on the **My Company Tags** tab.
4. Click **Delete** under the **Actions** column next to the tag to delete it.

   Note:

   When you delete a tag, the tag is also deleted from any resource to which it was assigned.
5. Click **OK** to confirm.

Importing Tags for Virtual Machines

You can import a **CSV** file with tag assignments into the VMDB. For the import to be successful, be aware of the following:

- The file must be in the following format, with one line for each virtual machine. One virtual machine per tag must be on a separate line even if you are assigning multiple tags of the same category.
- You must use the display names of the category and the display name for the tag for the import to work.

```
name,category,entry
evm2,Provisioning Scope,All
evm2,Exclusions,Do not Analyze
evm2,EVM Operations,Analysis Successful
rhel6,Department,Presales
rhel6,Department,Support
```

Importing Tags for a Virtual Machine from a CSV File

To import tags for a virtual machine from a CSV file:

1. Make sure the **CSV file** is in the required format.
2. Browse to **Settings > Application Settings**.
3. Click on the **Settings** accordion, then **Region**.
4. Select the **Tags** tab, then click on the **Import Tags** tab.
5. Click **Choose file** to go to the location where the file is located.
6. Click **Upload**.

   Note:

   If there are any problems with the file, such as an incorrect column name, unknown virtual machine, unknown tag, or multiple values for a tag that should have only one, an error message will appear in the console for those records.
7. Click **Apply**.

Importing Custom Values for Virtual Machines and Hosts

You can import a **CSV** file with asset tag information into the VMDB for a virtual machine or import custom values for hosts. For the import to be successful, the file must be in the following format, with one line for each virtual machine or host.

- There are two columns.
- The first line of the file must have the column names as shown below.
- The column names are case sensitive.
- Each value must be separated by a comma.

Virtual Machine Import Example

```
name,custom_1
Ecommerce,665432
Customer,883452
SQLSrvr,1090430
Firewall,8230500
```

For virtual machines, the value for custom_1 will show in the **VM Summary** page as the **Custom Identifier** page as the **Custom Identifier** in the **Properties** area. All of the custom values will show in the **Custom Fields** area.

Host Import Example

```
hostname,custom_1,custom_2
esx303.galaxy.local,15557814,19948399
esxd1.galaxy.local,10885574,16416993
esxd2.galaxy.local,16199125,16569419
```

For hosts, the value for custom_1 will show in the **Host Summary** page as the **Custom Identifier** in the **Properties** area. All of the custom values will show in the **Custom Fields** area.

Importing Asset Tags for a Virtual Machine from a CSV File

To import asset tags for a virtual machine from a CSV file

1. Make sure the **CSV file** is in the required format.
2. Browse to **Settings > Application Settings**.
3. Click on the **Settings** accordion, then **Region**.
4. Select the **Tags** tab, then click on the **Import Variables** tab.
5. Select the type of custom variable you want to import, either **Host** or **VM**.
6. Click **Choose file** to go to the location where the custom variable file is located.
7. Click **Upload**.

   Note:

   If there are any problems with the file, such as an incorrect column name, unknown virtual machine or host, a message appears.
8. Click **Apply**.

Tag Mapping

You can map labels from external providers to tags within Infrastructure Automation.

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then select the **Region**.
3. Select the **Tags** tab, then click on the **Map Tags** tab.
4. Click **Add** to create a tag map.
5. In the **Choose a resource entity and label**, select the **Entity** and enter a **Resource label**, then enter a **Tag Category**.
6. Click **Add**.

Customizing the Help Menu

Infrastructure Automation allows administrators to customize the help menu. Use this feature to define menu labels, URLs and how each window opens for users.

Note:

Any change to the help menu will take effect upon a full page reload.

Customize the help menu using the following steps:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then **Region**.
3. Click on the **Help Menu** tab.
4. Provide custom **Menu item labels** and an associated **URL** for each. Define how each window should open by selecting from the options in the **Open in** menu.
5. Click **Submit**.

Profiles

Creating an Analysis Profile

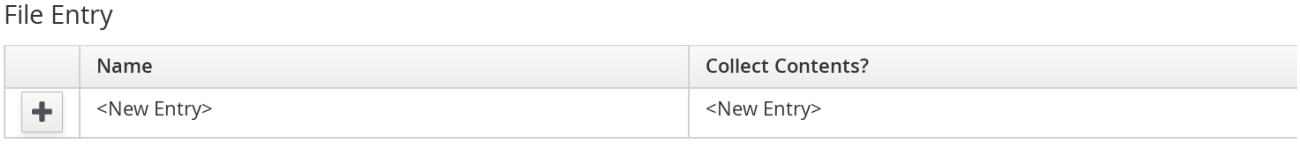
You can create an analysis profile by referring to the sample profiles provided in the console. You can copy the sample profile or create a new one.

Creating a Host Analysis Profile

To create a host analysis profile:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Analysis Profiles**.
3. Click **Configuration**, and ⊕ (**Add Host Analysis Profile**).
4. In the **Basic Information** area, type in a **Name** and **Description** for the analysis profile.

5. Click **File** to collect information about a file or group of files.
6. From the **File Entry** area, click ⊕ (**Click to add a new entry**) to add a file or group of files.



   - Check **Collect Contents** to not only check for existence, but also gather the contents of the file. If you do this, then you can use the contents to create policies in Infrastructure Automation Control.
7. Click **Event Log** to specify event log entries to collect.

8. From the **Event Log Entry** area, click **Click to add a new entry** to add a type of event log entry. Type in a **Name**. You can type in a specific message to find in **Filter Message**. In **Level**, set the value for the level of the entry and above. Specify the **Source** for the entry. Finally, set the # number of days that you want to collect event log entries for. If you set this to 0, it will go as far back as there is data available.

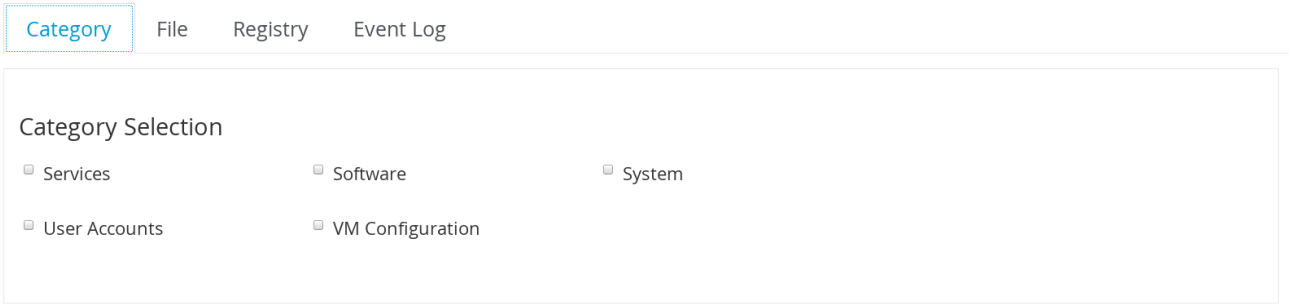
9. Click **Add**.

Creating a Virtual Machine Analysis Profile

To create a virtual machine analysis profile:

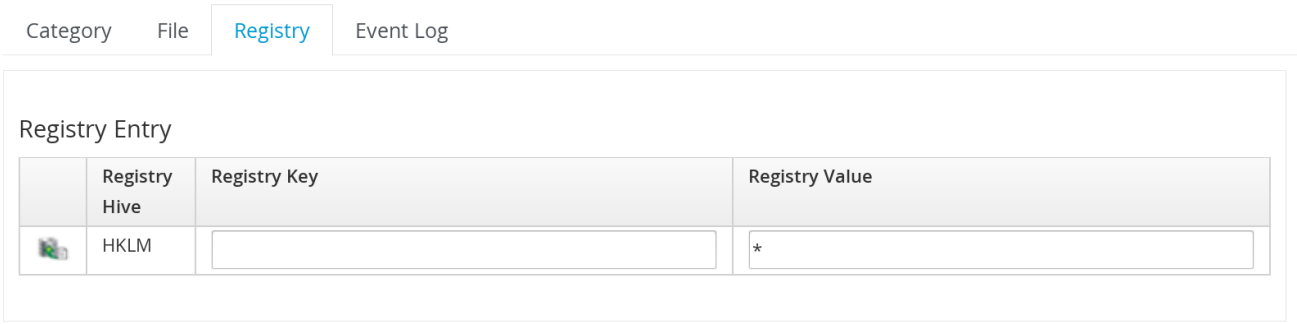
1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Analysis Profiles**.
3. Click **Configuration**, and **Add VM Analysis Profile**.
4. In the **Basic Information** area, type in a **Name** and **Description** for the analysis profile.

5. You begin in the **Category** tab. From the **Category Selection** area, check the categories you want to collect information for. This is available for virtual machine profiles only.



6. Click the **File** tab to collect information about a file or group of files.
7. From the **File Entry** area, click **Add this entry** to add a file or group of files, then type a name. For virtual machines, specify the file to check for. Check the box under **Collect Contents** if you want to collect the file contents as well. The files can be no larger than 1 MB.

8. Click the **Registry** tab to collect information on a registry key.
9. From the **Registry Entry** area, type your **Registry Key** and **Registry Value**. To evaluate whether a registry key exists or does not exist on a virtual machine, without providing a value, type * in the **Registry Value** field. Then, you do not need to know the registry value to collect the keys. This is available for virtual machine profiles only.



10. Click **Event Log** to specify event log entries to collect.
11. From the **Event Log Entry** area, complete the fields to add a type of event log entry. You can type in a specific message to find in **Filter Message**. In **Level**, set the value for the level of the entry and above. Specify the **Source for the entry**. Finally, set the # (number) of days that you want to collect event log entries for. If you set this to 0, it will go as far back as there is data available.

12. Click **Add**.

Editing an Analysis Profile

To edit an analysis profile:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Analysis Profiles**.
3. Check the analysis profile you want to edit.
4. Click **Edit the selected Analysis Profiles**.
5. Make any changes.
6. Click **Save**.

The changes are added to the analysis profile. The virtual machines or hosts must be re-analyzed to collect the new or modified information.

Copying an Analysis Profile

To copy an analysis profile:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Analysis Profiles**.
3. Check the analysis profile you want to copy.
4. Click **Configuration**, then **Copy the selected Analysis Profiles**.
5. Type a new **Name** and **Description**.
6. Make required changes.
7. Click **Add**.

Setting a Default Analysis Profile

If you want to set an analysis profile to be used for all virtual machines, you can create a default profile.

To create a default analysis profile:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Analysis Profiles**.
3. Click on the analysis profile you want to set as the default.
4. Click **Configuration**, then **Edit the selected Analysis Profile**.
5. For a virtual machine profile, enter `default` in lower case in **Name**. For a host profile, enter `host default`.

6. Click **Save**.

Zones

You can organize your Infrastructure Automation Infrastructure into zones to configure failover and isolate traffic. A provider that is discovered by a server in a specific zone gets monitored and managed in that zone. All jobs, such as a SmartState Analysis or VM power operation, dispatched by a server in a specific zone can get processed by any Infrastructure Automation appliance assigned to that same zone.

Zones can be created based on your own environment. You can make zones based on geographic location, network location, or function. When first started, a new server is put into the default zone.

Suppose you have four Infrastructure Automation appliances with two in the East zone, appliances A and B, and two in the West zone, appliances C and D. VC East is discovered by one of the Infrastructure Automation appliances in the Infrastructure Automation Eastern zone. If Appliance A dispatches a job of analyzing twenty virtual machines, this job can be processed by either Appliance A or B, but not C or D.

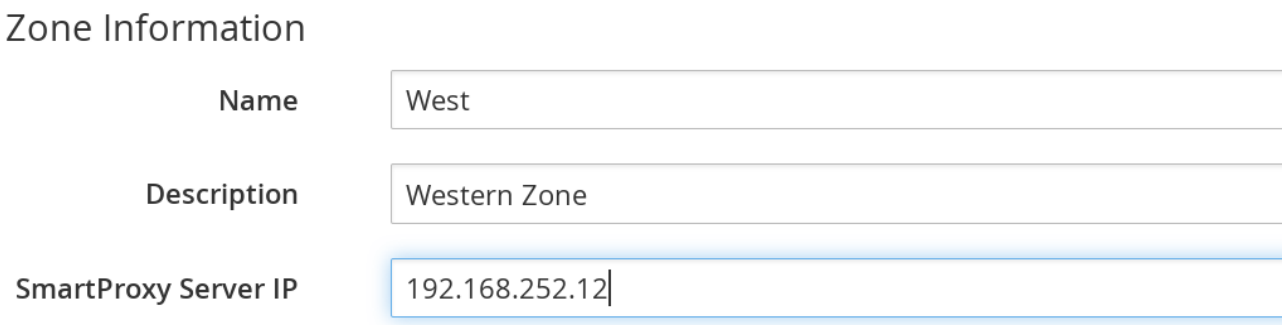
Note:

Only users assigned the super administrator role can create zones. There must always be at least one zone. The **Default Zone** is provided and cannot be deleted.

Creating a Zone

To create a zone:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click **Configuration**, and **Add a new Zone** to create a zone.
4. In the **Zone Information** area, type in a **Name** and **Description** for the new zone.



5. Use **SmartProxy Server IP** to specify the IP address of the server that you want SmartProxies installed in this zone to report to. If this is not set, then the IP address of the server that deployed the SmartProxy is used. This does not apply to embedded SmartProxies.
6. Optionally, you can configure **NTP servers** for the entire zone in the NTP Servers area. These settings will be used if the NTP servers have not been set for the appliance in the menu: **Operations > Server** page.
7. In the **Settings** area, set the number for **Max Active VM Scans**. The default is `Unlimited`.
8. Click **Save**.

Deleting a Zone

To delete a zone:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone you want to remove.

   Note:

   You cannot delete a zone if there are servers assigned to it.
4. Click **Configuration**, then click **Delete this Zone**.
5. Click **OK** to confirm.

Editing a Zone

To edit a zone:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone you want to edit.
4. Click **Configuration**, then click **Edit this Zone**.
5. Complete the required changes.
6. Click **Save**.

Adding SmartProxy Affinity to a Zone:

Enable SmartProxy Affinity for zones containing servers with the SmartProxy role to run a SmartState Analysis.

To add SmartProxy Affinity to a zone:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone in which you want to enable SmartProxy Affinity.
4. Click the **SmartProxy Affinity** tab and click the appropriate server.
5. Click **Save**.

Servers

Server settings enables you to control how each Infrastructure Automation server operates including authentication, logging, and email. If you have multiple servers in your environment that are reporting to one central VMDB, then you can edit some of these settings from the console by specifying which server you want to change.

Note:

The server selection options are only available if you have multiple servers sharing one VMDB.

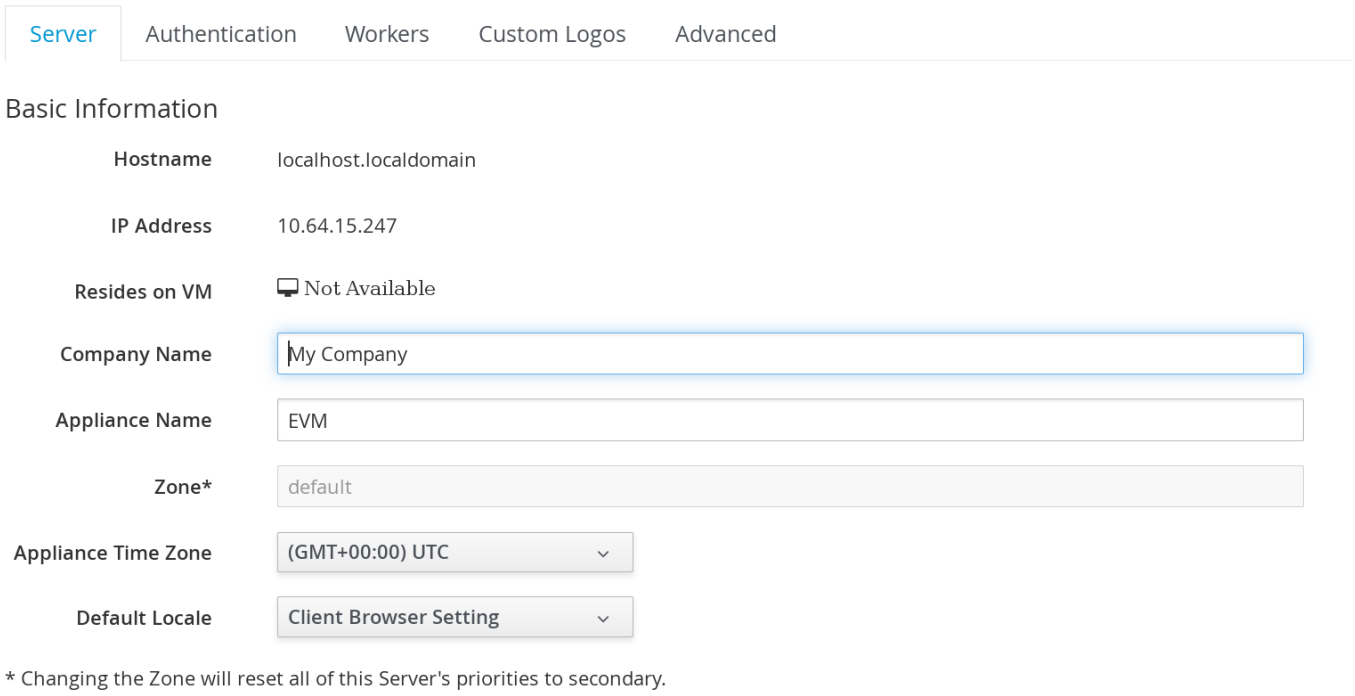
Changing Server Settings

To change server settings:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone where the Infrastructure Automation server is located.
4. In the **Servers** area, click on the Infrastructure Automation server.

5. Click **Server**.
6. Make any required changes.
7. Click **Save**.

Basic Information Settings



- Use **Company Name** (maximum 20 characters) to customize the interface with your company's name. You will see the company name when you are viewing or modifying the tags of an infrastructure object or virtual machine.
- Specify the **Appliance Name** (maximum 20 characters) you want displayed as the appliance that you are logged into. You will see this in the upper right corner of the interface with the name of the consoles logged on user.
- Use **Zone** to isolate traffic and provide load balancing capabilities. Specify the zone that you want this Infrastructure Automation appliance to be a member of. At startup, the zone is set to default.
- Use **Appliance Time Zone** to set the time zone for this server.

  Note:

  This is the time zone used when created scheduled analyses. This is not the same as the **Time Zone** parameter, which is found by navigating to the settings menu, then **My Settings**, then exploring the **Display Settings** area, and is the time zone displayed in the console.
- Use **Default Locale** to specify the default language for this server.

Server Control Settings

A server role defines what a server can do. Red Hat recommends that Database Operations, Event Monitor, Reporting, Scheduler, SmartState Analysis, User Interface, Provider Inventory, Provider Operations, and Web Services be enabled on at least one server in each zone. These roles are enabled by default on all servers.

- Use **Default Repository SmartProxy** to set the SmartProxy from which you refresh your virtual machine repositories. This host must have access to your repositories to analyze its virtual machines.

Note:

- Only super administrators can change server roles.
- If you are using more than one Infrastructure Automation appliance, be sure to set this on all of the appliances.

Server Roles

Note:

- Server roles that are in an active/active high availability configuration *(load balancing and failover protection)* are active in more than one location; whereas, roles that are in an active/passive *(primary/secondary in the case of Infrastructure Automation)* high availability configuration *(failover protection)*, if more than one Infrastructure Automation server in a specific zone or region has this role, only one will be active *(primary)* at a time and a failover has to occur to the passive *(secondary)* appliance with that role.
- For information on region and zone diagnostics and server role priorities, see Diagnostics.

Server Role	Description	Zone or Region Aware	Primary/Secondary or Active/Active
Automation Engine	Use this role if you want to use this Infrastructure Automation server to process automation tasks.	N/A	Active/Active
Capacity and Utilization Coordinator	The Capacity & Utilization Coordinator role checks to see if it is time to collect data, somewhat like a scheduler. If it is time, a job is queued for the Capacity and Utilization Data Collector. The coordinator role is required to complete Capacity and Utilization data collection. If more than one Infrastructure Automation server in a specific zone has this role, only one will be active at a time.	Zone	Primary/Secondary
Capacity & Utilization Data Collector	The Capacity & Utilization Data Collector performs the actual collection of capacity and utilization data. This role has a dedicated worker, and there can be more than one Infrastructure Automation server with this role in a zone.	Zone	Active/Active
Capacity & Utilization Data Processor	The Capacity & Utilization Data Processor processes all of the data collected, allowing Infrastructure Automation to create charts. This role has a dedicated worker, and there can be more than one Infrastructure Automation server with this role in a zone.	Zone	Active/Active
Database Operations	Use Database Operations to enable this Infrastructure Automation server to run database backups or garbage collection.	Zone	Active/Active
Embedded Ansible	This role is disabled by default. The Embedded Ansible role supports Ansible Automation Inside functionality. Enable this role to configure playbook repositories and run playbooks natively to back service catalog items. **NOTE**: Enable the Provider Inventory server role in the same zone as the Embedded Ansible server role to ensure proper functionality.	Region	Primary/Secondary
Event Monitor	This role is enabled by default and provides the information shown in timelines. The Event Monitor is responsible for the work between the Infrastructure Automation server and your providers. It starts 2 workers for each provider. One worker, the monitor, is responsible for maintaining a connection to a provider, catching events, and putting them on the Infrastructure Automation message queue for processing. The second worker, the handler, is a message queue worker responsible for delivering only those messages for a provider. You should have at least one of these in each zone.	Zone	Primary/Secondary
Git Repository	The Git Repositories Owner server role supports importing domains into automate from a git repository. This feature is available from the **Automate > Import/Export** screen in the Infrastructure Automation user interface.	Region	Primary/Secondary
Notifier	Use this role if you will be using Infrastructure Automation Control or Automate to forward SNMP traps to a monitoring system or send e-mails. See Creating an Alert for details on creating SNMP alerts. If more than one Infrastructure Automation server in a specific region has this role, only one will be active at a time.	Region	Primary/Secondary
Provider Inventory	This role is enabled by default. This role is responsible for refreshing provider information including EMS, hosts, virtual machines, and clusters, and is also responsible for capturing datastore file lists. If more than one Infrastructure Automation server in a specific zone has this role, only one will be active at a time. Required in the same zone as a Infrastructure Automation appliance with the Embedded Ansible role enabled.	Zone	Primary/Secondary
Provider Operations	This role is enabled by default. This role sends stop, start, suspend, shutdown guest, clone, reconfigure, and unregister to the provider, directly from the console or through a policy action if you have Infrastructure Automation Control. More than one Infrastructure Automation server can have this role in a zone.	Zone	Active/Active
RHN Mirror	An appliance with RHN Mirror enabled acts as a server containing a repository with the latest Infrastructure Automation packages. This also configures other appliances within the same region to point to the chosen RHN Mirror server for updates. This provides a low bandwidth method to update environments with multiple appliances.	N/A	Active/Active
Reporting	This role is enabled by default. The Reporting role specifies which Infrastructure Automation servers can generate reports. If you do not have a Infrastructure Automation server set to this role in a zone, then no reports can be generated in that zone. You should have at least one of these in each zone.	Zone	Active/Active
Scheduler	This role is enabled by default. The Scheduler sends messages to start all scheduled activities such as report generation and SmartState analysis. This role also controls all system schedules such as capacity and utilization data gathering. One server in each region must be assigned this role or scheduled Infrastructure Automation events will not occur. If more than one Infrastructure Automation server in a specific region has this role, only one will be active at a time.	Region	Primary/Secondary

Server Role	Description	Zone or Region Aware	Primary/Secondary or Active/Active
SmartProxy	Enabling the SmartProxy role turns on the embedded SmartProxy on the Infrastructure Automation server. The embedded SmartProxy can analyze virtual machines that are registered to a host and templates that are associated with a provider. To provide visibility to repositories, install the SmartProxy on a host from the Infrastructure Automation console. This SmartProxy can also analyze virtual machines on the host on which it is installed. Enabling the SmartProxy role on an appliance requires selecting the SmartProxy Affinity for a zone to run a SmartState Analysis. By default, no selections are enabled under SmartProxy Affinity.	Zone	Active/Active
SmartState Analysis	This role is enabled by default. The SmartState Analysis role controls which Infrastructure Automation servers can control SmartState Analyses and process the data from the analysis. You should have at least one of these in each zone.	Zone	Active/Active
User Interface	This role is enabled by default. The Web Services role must also be enabled with this role to log into the user interface, as the User Interface role queries the API to receive tokens for login. Uncheck User Interface if you do not want users to be able to access this Infrastructure Automation server using the Infrastructure Automation console. For example, you may want to turn this off if the Infrastructure Automation server is strictly being used for capacity and utilization or reporting generation. More than one Infrastructure Automation server can have this role in a zone.	Zone	Active/Active
Web Services	This role is enabled by default. The Web Services roles provides API access and must be enabled if the User Interface role is enabled to log into the user interface. You can also enable the Web Services role to provide API-only access to the server. Uncheck Web Services to stop this Infrastructure Automation server from acting as a web service provider. More than one Infrastructure Automation server can have this role in a zone.	N/A	Active/Active
Websocket	This role enables starting or stopping websocket workers required for proxying remote consoles.	N/A	Active/Active

VMware Console Settings

If you are using the Infrastructure Automation control feature set, then you have the ability to connect to a Web console for virtual machines that are registered to a host. To use this feature, you must have VNC installed, VMware's WebMKS SDK enabled in Infrastructure Automation, or the VMRC native desktop application installed for your environment.

Note:

See vendor documentation to ensure you are installing appropriate applications for your virtual infrastructure. Once you have installed the required software, you must specify its version in the Infrastructure Automation configuration settings.

- If you select **VNC**, type in the port number used. This port must be open on the target virtual machine and the VNC software must be installed there. On the computer that you are running the console from, you must install the appropriate version of Java Runtime if it is not already installed.
- If you select **VMware WebMKS**, select the appropriate version.
- If using **VMware VMRC** desktop application, be sure that you have fulfilled the requirements for your vCenter version. The correct version of the VMRC desktop application from VMware must be installed on the client computer. To do this, log into the vCenter Web Service and attempt to open a virtual machine console. The vSphere Web Client must be installed on vCenter version 5, and the provider must be registered to it. For vCenter version 4, the VMware vCenter Management Webservices must be running.

NTP Servers Settings In the NTP Servers area, you can

specify the NTP servers to use as source for clock synchronization here. The NTP settings specified here will override Zone NTP settings. Enter one NTP server hostname or IP address in each text box.

Configuring SNMP

You can use Simple Network Management Protocol (SNMP) traps to send alerts for various aspects of a Infrastructure Automation environment.

Requirements

- Configure your SNMP management station to accept traps from Infrastructure Automation appliances. Consult your management station's documentation.
- Each appliance that could process SNMP traps must have the **`snmpd`** and **`snmptrapd`** daemons running.
- The region where the appliances are located must have the **`Notifier`** role enabled and the failover role priority set.

To enable the **`snmpd`** and **`snmptrapd`** daemons

1. Access each SNMP processing appliance using SSH.
2. Set the SNMP daemons to run on start up:

   ```
   # chkconfig --level 2345 snmpd on
   # chkconfig --level 2345 snmptrapd on
   ```

3. The daemons run automatically when the appliance is restarted, but must be started manually now.

   ```
   # service snmpd start
   # service snmptrapd start
   ```

To enable the notifier role:

1. Access each SNMP processing appliance using their web interfaces.
2. Browse to **Settings > Application Settings**.
3. Click on the **Settings** accordion, then click **Zones**. Select the zone where the EVM server is located, and select the EVM server.
4. In the **Server Control** area, set the **Notifier** server role option to **`ON`**.
5. Click **Save**.

To set the failover priority role:

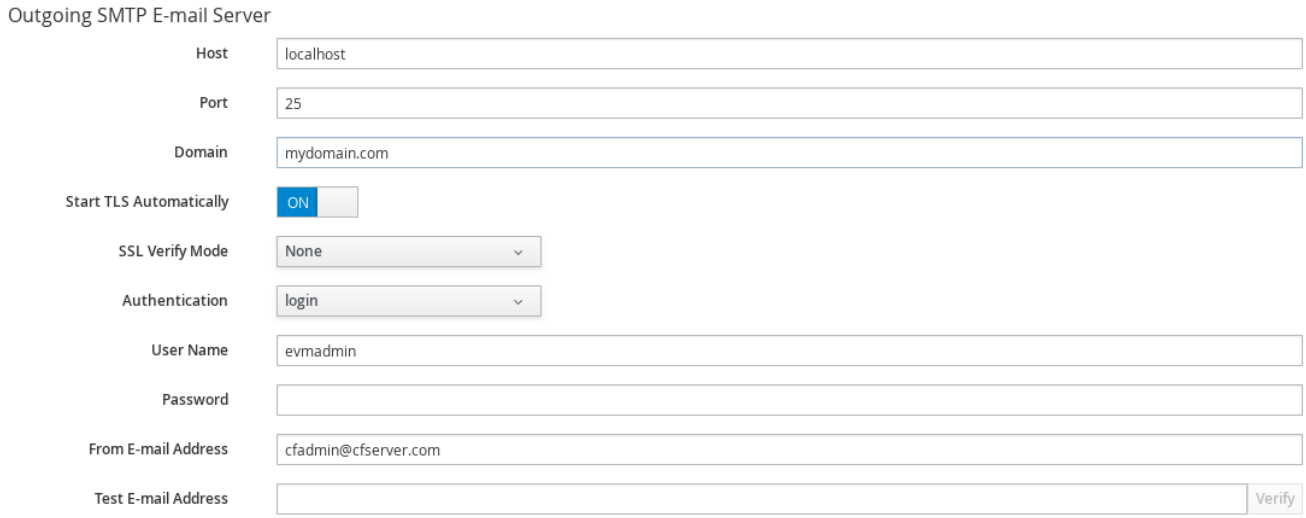
1. Browse to **Settings > Application Settings** then click the **Diagnostics** accordion.
2. Select the zone where the EVM server is located.
3. Click **Roles by Servers** or **Servers by Roles** to view your servers.
4. In the **Status of Roles for Servers** in **Zone Default Zone** area, click the role that you want to set the priority for.
5. Click **Configuration**, and **Promote Server** to make this the primary server for this role.

Outgoing SMTP Email Settings

To use the email action in Infrastructure Automation, set an email address to send emails from.

Note:

To be able to send any emails from the server, you must have the Notifier server role enabled. You can test the settings without the role enabled.



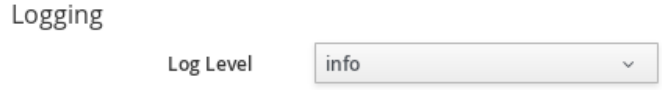
- Use **Host** to specify the host name of the mail server.
- Use **Port** to specify the port for the mail server.
- Use **Domain** to specify the domain name for the mail server.
- Set **Start TLS Automatically** on `ON` if the mail server requires TLS.
- Select the appropriate **SSL Verify Mode**.
- Use the **Authentication** drop down to specify if you want to use `login`, `plain`, or no authentication.
- Use **User Name** to specify the user name required for login authentication.
- Use **Password** to specify the password for login authentication.
- Use **From E-mail Address** to set the address you want to send the email from.
- Use **Test E-mail Address** if you want to test your email settings. Click **Verify** to send a test email.

Web Services Settings

Web services are used by the server to communicate with the SmartProxy.

- Set **Mode** to invoke to enable 2-way Web services communication between the Infrastructure Automation appliance and the SmartProxy. Set **Mode** to disabled to use Web services from the SmartProxy to the Infrastructure Automation appliance only. When the Infrastructure Automation appliance has work for the SmartProxy, the work will be placed in a queue in the VMDB. The work will be completed either when the Infrastructure Automation appliance is able to contact the SmartProxy or when the next SmartProxy heartbeat occurs, whichever comes first.
- If **Web Services** are enabled, you have the option to use **ws-security**.

Logging Settings



- Use **Log Level** to set the level of detail you want in the log. You can select from **fatal**, **error**, **warn**, **info**, and **debug**. The default setting is 'info'.

Custom Support URL Settings

- Use **URL** to specify a specific URL that you want to be accessible from the **About Product Assistance** area.
- Use **Description** to set a label for the **URL**.

Authentication

Use the **Authentication** tab to specify how you want users authenticated on the console. You can use the VMDB or integrate with LDAP, LDAPS, Amazon, or an external IPA server.

Changing Authentication Settings

To change authentication settings:

1. Browse to **Settings > Application Settings**.
2. Click the **Settings** accordion, then click **Zones**.
3. Click the zone where the server is located.
4. Click the server.
5. Click the **Authentication** tab.
6. Use **Session Timeout** to set the period of inactivity before a user is logged out of the console.
7. Set the authentication method in **Mode**.
8. Click **Save**.

Workers

Use the Workers page to specify the number of workers and amount of memory allowed to be used for each type.

Note:

Only make these changes when directed to by Red Hat Support.

Changing Settings for a Worker

To change the settings for a worker

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone where the server is located.
4. Click on the server.
5. Click **Workers**.
6. Go to the type of worker you have been directed to change.
7. If applicable, change Count or Memory Threshold using the dropdown boxes.
8. Click **Save**.

Database

Use the Database page to specify the location of your Virtual Machine Database (VMDB) and its login credentials. By default, the type is PostgreSQL on the Server.

Note:

The server may not start if the database settings are changed. Be sure to validate your new settings before restarting the server.

Changing a Database Setting

To change a database setting:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone where the server is located.
4. Click on the server.
5. Click the **Database** tab.
6. In the **Database** area, select the **Type** of database. You can select from **External Database on another CFME appliance**, **External Postgres Database**, and **Internal Database on this CFME Appliance**.
    - Use **Hostname** to specify the IP address or hostname of the external database server.
    - Use **Database Name** to specify the name of your VMDB.
    - Specify the **User Name** to connect to the VMDB.
    - Use **Password** and **Verify Password** to specify the password for the user name.
7. Click **Validate** to check the settings.
8. Click **Save**.
9. Click **OK** to the warning that the server will restart immediately after you save the changes.

During the restart, you are unable to access the server. When the restart is complete, the new database settings are in effect.

Customization and Logos

Custom Logos

Use **Custom Logos** to display your own logo in the corner of the Infrastructure Automation user interface and on the login screen. Use the procedures below to upload a custom logo to the user interface, and to customize the login background and login panel text on the user interface.

Note:

- If you have upgraded from an earlier Infrastructure Automation version and your custom logo was already in use before migration, although your logo image file is still in place in **`vmdb/public/upload`** you may have to uncheck and recheck the option to **Use Custom Logo Image** to re-enable displaying your custom logo. See Uploading a Custom Logo to the User Interface for the procedure on how to access the **Use Custom Logo Image** option, or if you want to upload another custom logo to the user interface and customize the login background image and login panel text.
- Additionally, ensure the option to use configuration settings for the tenant under **Access Control** is set to **`Yes`**; see Displaying the Custom Configuration Settings for the procedure on how to set the configuration settings.

Uploading a Custom Logo to the User Interface

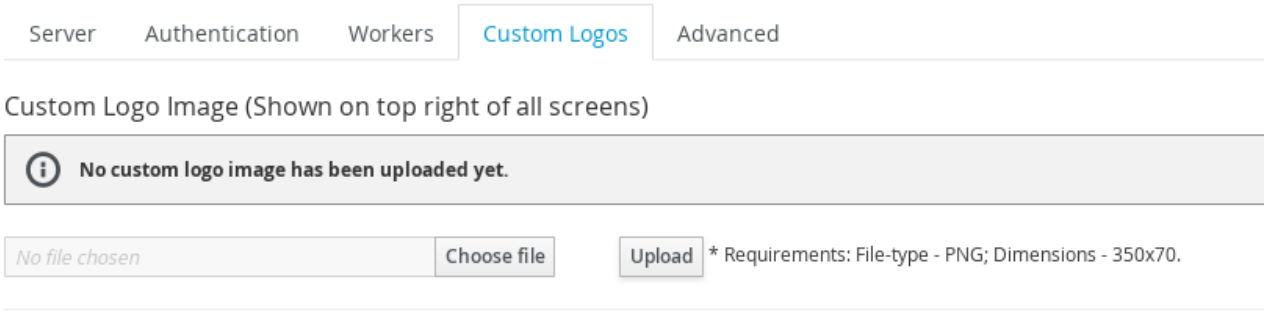
Note:

Make sure the desired logo is accessible from the computer where you are running the Infrastructure Automation user interface. The file must be in portable network graphics (png) format with dimensions of 350 px x 70 px.

To upload a custom logo to the user interface:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone where the Infrastructure Automation server is located.
4. Click on the server.

5. Click the **Custom Logos** tab.



6. In **Custom Logo Image (Shown on top right of all screens)**, click **Choose file** to go to the location where the logo file is located.
7. Click **Upload**. The icon is displayed above the file name box, and an option is shown to use the logo.
8. Check **Use Custom Logo Image** to add the logo to your user interface.
9. Click **Save**.

Note:

To enable displaying your custom logo, ensure the option to use configuration settings for the tenant under **Access Control** is set to `Yes`. See simpara_title for the procedure on how to set the configuration settings.

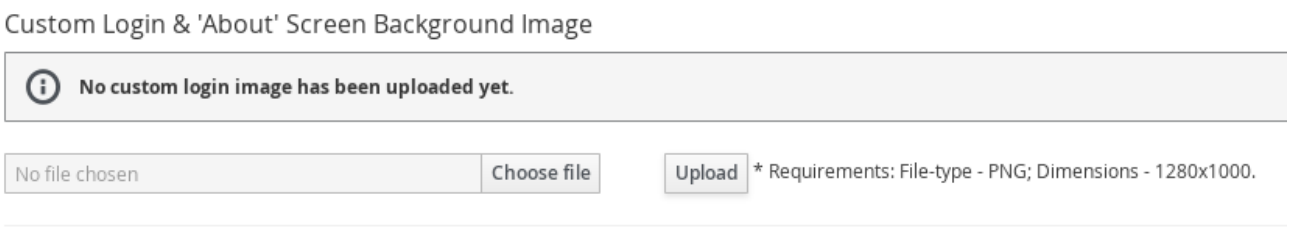
Customizing the Login Background

Note:

Make sure the background image that you want to use is accessible from the computer where you are running the user interface. The file must be in PNG format with dimensions of 1280 px x 1000 px.

To customize the login background:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone where the server is located.
4. Click on the server.
5. Click the **Custom Logos** tab.
6. In **Custom Login & 'About' Screen Background Image**, click **Choose file** to go to the location where the background image file is located.



7. Click **Upload**. The icon is displayed above the file name box, and an option is shown to use the logo.
8. Check **Use Custom Login Background Image** to add the background image to the login screen of the user interface.
9. Click **Save**.

Customizing the Login Panel Text

To customize the login panel text:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone where the server is located.

4. Click on the server.
5. Click the **Custom Logos** tab.
6. In **Custom Login Panel Text**, enter the text that you want to display on the login screen.
7. Click **Use Custom Login Text** to switch it to **`Yes`**.

8. Click **Save**.

Displaying the Custom Configuration Settings

To enable displaying your custom logo in the corner of the Infrastructure Automation user interface and on the login screen:

1. Browse to **Settings > Application Settings**.
2. Click the **Access Control** accordion.
3. Click **Tenants**, then click **My Company**.
4. Click (**Configuration**), then click (**Edit this item**).

5. Click **Use Configuration Settings** to switch it to **`Yes`**.
6. Click **Save**.

Advanced Settings

You may be instructed by Red Hat to edit some configuration settings manually. This feature is available for a limited number of options and can only be used by users assigned the super administrator role. Changing settings using this procedure may disable your Infrastructure Automation server.

- Only make manual changes to your configuration files if directed to do so by Red Hat. - Infrastructure Automation provides the ability to edit advanced configuration settings for all appliances through the **Advanced** tab from the user interface. You can also specify advanced settings for entire zones by clicking the **Advanced** tab at the zone level, with the advanced settings applied to individual appliances in that zone overriding zone-level settings when set.

Editing Configuration Files Manually

To edit configuration files manually:

1. Click **Configuration**.
2. Click on the **Settings** accordion, then click **Zones**.
3. Click the zone where the server is located.
4. You can click the **Advanced** tab at the zone level to set advanced configuration settings for the entire zone. Or, you can click on the server, and then click the **Advanced** tab to set advanced configuration settings at the server (appliance) level, which will override zone-level settings.
5. Make the required changes to the configuration file.
6. Click **Save**.

Table: authentication

Parameters	Description
amazon_key	If using Amazon for the authentication mode, specify your Amazon Key. This is the same as Amazon Access Key in Configuration-Operations-Server-Amazon Settings in the appliance console. Default: blank
amazon_secret	If using Amazon for the authentication mode, specify your Amazon Secret. This is the same as Amazon Secret Key in Configuration-Operations-Server-Amazon Settings in the appliance console. Default: blank
basedn	If using ldap for the authentication mode, specify your Base DN. This is the same as Base DN in Configuration-Operations-Server-LDAP Settings in the appliance console. Default: blank
bind_dn	The user name to bind to the LDAP server. This user must have read access to all users and groups that will be used for Infrastructure Automation authentication and role assignment. This is the same as Bind DN in Configuration-Operations-Server-LDAP Settings in the appliance console. Default: blank
bind_pwd:	The password for the bind_dn user. This is the same as Bind Password in Configuration-Operations- Server-LDAP Settings in the appliance console. Default: blank
get_direct_groups	Use this to get the LDAP roles from the LDAP users' home forest. This is the same as Get Roles from Home Forest in the Authentication page for the Infrastructure Automation Server. Default: true
group_memberships_max_depth	When traversing group memberships in the LDAP directory it will stop at this value. Default: 2
ldaphost	Use ldaphost to specify the fully qualified domain name of your LDAP server. This is the same as LDAP Host Name in Configuration-Operations-Server-LDAP Settings in the appliance console. Default: blank
ldapport	Specify the port of your LDAP server. This is the same as LDAP Port in Configuration-Operations- Server-LDAP Settings in the appliance console. Default: 389
mode	Use database to use the VMDB for security. Use ldap or ldaps to use directory services. This is the same as Mode in Configuration-Operations-Server-Authentication in the appliance console. Default: database
user_type	Use userprincipalname to type the user name in the format of user@domainname. Use mail to login with the user's e-mail address. Use dn-cn for Distinguished Name (CN=<user>) or dn-uid Distinguished Name (UID=<user>) to use just the user name, but be sure to enter the proper user_suffix for either one. This is the same as User Type in Configuration-Operations-Server-LDAP Settings in the appliance console. Default: userprincipalname
user_suffix	Domain name to be used with user_type of dn-cn or dn-uid. This is the same as User Suffix in Configuration-Operations-Server-LDAP Settings in the appliance console. Default: blank

Table: coresident_miqproxy

Parameters	Description
use_vim_broker	Specify if you want the coresident SmartProxy to use a shared connection through the VIM broker to communicate with the VC or ESX host for SmartState Analysis. If it is disabled, then each SmartProxy SmartState Analysis would create its own connection. Default: true
concurrent_per_ems	Specify the number of co-resident SmartProxy SmartState Analyses that can be run against a specific management system at the same time. Default: 1
concurrent_per_host	Specify the number of co-resident SmartProxy SmartState Analyses that can be run against a specific host at the same time. Default: 1
scan_via_host	If you change scan_via_host to false, Infrastructure Automation will use the Management System to scan which is limited by the concurrent_per_ems setting instead of the concurrent_per_host setting. Note this will greatly increase traffic to the Management System. Default: true

Table: ems_refresh

Parameters	Description
capture_vm_created_on_date	Set to false to turn off historical event retrieval. Set to true to turn on. By setting the flag to true Infrastructure Automation will try to set the "ems_created_on" column in the vms table after an ems refresh for new VMs and any VMs with a nil "ems_created_on" value. Infrastructure Automation looks at event information in our database as well as looking up historical event data from the management system. This is optional since the historical lookup could timeout. Default: false
collect_advanced_settings	Set to false if you do not want to collect advanced Virtual Machine settings during a management system refresh. This will increase the speed of the refresh, but less data will be collected. If the parameter is not listed, then the value is true. Default: true
ec2	
get_private_images	For EC2 refreshes only; whether or not to retrieve private images. Default: true
get_public_images	For EC2 refreshes only; whether or not to retrieve public images. Default: false. Warning: setting get_public_images to true loads several thousand images in the VMDB by default and may cause performance issues.
get_shared_images	For EC2 refreshes only; whether or not to retrieve shared images. Default: true
public_images_filters	For EC2 refreshes only; a filter to reduce the number of public images. Default: all public images

Parameters	Description
ignore_terminated_instances	For EC2 refreshes only; whether or not to ignore terminated instances. Default: true
full_refresh_threshold	The number of targeted refreshes requested before they are rolled into a full refresh. For example, if the system and/or the user target a refresh against 7 VMs and 2 Hosts (9 targets), when the refresh actually occurs it will do a partial refresh against those 9 targets only. However, if a 10th had been added, the system would perform a full EMS refresh instead. Default: 100
raise_vm_snapshot_complete_if_created_within:	Raises vm_snapshot_complete event for a snapshot being added to VMDB only if the create time in Virtual Center is within the configured period of time. This prevents raising events for old snapshots when a new VC is added to Infrastructure Automation. Default: 15.minutes
refresh_interval	Scheduler does a periodic full EMS refresh every refresh_interval. Default: 24.hours

Table: host_scan

Parameters	Description
queue_timeout	Time period after which a host SmartState analysis will be considered timed out. Default: 20.minutes

Table: log

Parameters	Description
level	Specify the required level of logging for the Infrastructure Automation appliance. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. This is the same as Log Level in Configuration-Operations-Server-Logging in the appliance console and applies immediately to the evm log. Default: info
level_aws	Specify the level of logging for Amazon Web Services communications. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. This applies to the aws log. Default: info
level_aws_in_evm	Specify what level of Amazon Web Services communication log should be also shown in evm log. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. Default: error
level_fog	Specify the level of logging for Fog communications. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. This applies to the fog log. Default: info
level_fog_in_evm	Specify what level of Fog communication log should be also shown in evm log. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. Default: error
level_rails	Specify the level of logging for Rails. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. Once changed, this applies immediately to the production log. Default: info
level_rhevm	Specify the level of logging for Red Hat communications. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. This applies to the rhevm log. Default: warn
level_rhevm_in_evm	Specify what level of Red Hat communication log should be also shown in evm log. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. Default: error
level_vim	Specify the level of logging for VIM (communication with VMware ESX and Virtual Center). Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. This applies to the vim log. Default: warn
level_vim_in_evm	Specify what level of vim logging should be also shown in the evm log. Possible levels from most detailed to least detailed are: debug, info, warn, error, fatal. Default: error

Table: db_stats

Parameters	Description
enabled	Specify if you want to keep track of the number of queries, size of queries, number of responses, size of response, min/max for each, number of established connections at for each server process. This information will show in the EVM log. Default: false
log_frequency	How frequently in seconds the process will log the database statistic in seconds. Default: 60

Table: callsites

Parameters	Description
enabled	Specify if you want keep track of the code that is accessing the database. Enabling call sites will decrease performance because of the amount of information tracked. The db_stats: enabled parameter must be set to true to use this. Default: false
depth	Specify how many levels in the call stack to track for each database access. Default: 10
min_threshold	Do not keep track of code that does not access the database this many times per log_frequency. Default: 10
path	Set the path for the Infrastructure Automation appliance log. This is the same as Log Path in Configuration-Operations- Server-Logging in the appliance console. Default: If no value is present, the path is /var/www/miq/vmdb/log.
line_limit	Limit how many characters are retained in a single log line. 0 means no limit. Default: 0

Table: collection

Parameters	Description
ping_depot	Whether to use TCP port ping to the log depot before performing log collection. Default: true
ping_depot_timeout	Specify how long in seconds to wait for response from log depot before deciding that the TCP port ping failed. Default: 20

Parameters	Description
current	When collecting logs, specifies what is considered current logging as opposed to archived logging. Default: :pattern: log/*.log log/apache/*.log log/*.txt config/* /var/lib/pgsql/data/*.conf /var/lib/pgsql/data/log/* /var/log/syslog* /var/log/daemon.log* /etc/default/ntp* /var/log/messages* /var/log/cron* BUILD GUID VERSION
archive	Specifies what is considered archived logging. The default pattern is blank which means *.gz files in the log directory.

Table: log_depot

Parameters	Description
uri	Specify the URI for the log depot. This is the same as the URI in **Configuration > Diagnostics > Collect Logs** in the appliance console. Default: blank
username	Specify the user name for the log depot. This is the same as the user ID in **Configuration > Diagnostics > Collect Logs** in the appliance console. Default: blank
password	Specify the password for the user for the log depot. This is the same as the password in **Configuration > Diagnostics > Collect Logs** in the appliance console. Default: blank

Table: performance

Parameters	Description
capture_threshold	
vm	Amount of time in minutes to wait after capture before capturing again. Default: 50.minutes
host	Amount of time in minutes to wait after capture before capturing again. Default: 50.minutes
ems_cluster	Amount of time in minutes to wait after capture before capturing again. Default: 50.minutes
storage	Amount of time in minutes to wait after capture before capturing again. Default: 120.minutes
capture_threshold_with_alerts	
host	Amount of time in minutes to wait after capture before capturing again. This value is used instead of capture_threshold for Hosts that have alerts assigned based on real time Capacity & Utilization data. Default: 20.minutes
ems_cluster	Amount of time in minutes to wait after capture before capturing again. This value is used instead of capture_threshold for clusters that have alerts assigned based on real time Capacity & Utilization data. Default: 50.minutes
vm	Amount of time in minutes to wait after capture before capturing again. This value is used instead of capture_threshold for VMs that have alerts assigned based on real time Capacity & Utilization data. Default: 20.minutes
concurrent_requests	
vm	Amount of time in minutes to wait after capture before capturing again. This value is used instead of capture_threshold for VMs that have alerts assigned based on real time Capacity & Utilization data. Default: 20.minutes
hourly	Number of concurrent VC requests to make when capturing hourly raw metrics. Default: 1
realtime	Number of concurrent VC requests to make when capturing real time raw metrics. Default: 20
history	
initial_capture_days	How many days to collect data for on first collection. Default: 0

Parameters	Description
Keep_daily_performances	How long to keep daily performance data in the VMDB. Default: 6.months
keep_realtime_performances	How long to keep realtime performance data in the VMDB. Default: 4.hours
keep_hourly_performances	How long to keep hourly performance data in the VMDB. Default: 6.months
purge_window_size	When the purge needs to delete rows which are older than the keep_realtime_performances, keep_hourly_performances, and keep_daily_performances values, this value sets how many rows to delete in each batch. For example, a value of 1000 will cause us to issue ten 1,000 row deletes. Default: 1000

Table: repository_scanning

Parameters	Description
defaultsmartproxy	Specify the SmartProxy for repository scanning. This is the same as Default Repository Smartproxy in Configuration-Operations- Server-VM Server Control in the appliance console. Default: blank

Table: server

Parameters	Description
case_sensitive_name_search	Specify if you want the search by name on configuration item screens to be case senstive. Default: false
company	Specify the label you want to use for your company's tagging. This is the same as Company Name in Configuration-Operations-Server-Basic Info. Default: "My Company"
custom_logo	Specify if you want to use a custom logo. This is the same as Use Custom Logo in Configuration-Custom Logo-Logo Selection. Default: false
events	
disk_usage_gt_percent	For Infrastructure Automation operational alerts, specify at what threshold the disk usage alerts will be triggered. Default: 80
heartbeat_timeout	How long to wait until the server heartbeat is considered timed out. if the timeout is exceeded, other appliances in the zone/region can vie for the roles active on the timed out Infrastructure Automation appliance. Default: 2.minutes
host	Infrastructure Automation Server's IP address. Default: blank
hostname	Infrastructure Automation Server's hostname. Default: localhost.localdomain
listening_port	Specify the port number on which the web server is listening. Note that this does not set the port that VMDB listens on. When deploying the SmartHost from the Infrastructure Automation appliance, it tells the SmartHost (miqhost) what port to talk to the VMDB on. Default: "443"
mks_version	Specify the version of the VMware MKS Plugin to use for the VM Console. This is the same as VMware MKS Plugin Version in Configuration-Operations- Server-VM Console. Default : 2.1.0.0
name	Set the name to display for the Infrastructure Automation appliance that you are logged on to in the appliance console. This is the same as appliance Name in Configuration-Operations- Server-Basic Information. Default : EVM
role	Specify the roles for this Infrastructure Automation Server, separated by commas without spaces. The possible values are automate, database_operations, ems_inventory, ems_metrics_collector, ems_metrics_coordinator, ems_metrics_processor, ems_operations, event, notifier, reporting, scheduler, smartproxy, smartstate, user_interface, web_services. This is the same as Server Roles in Configuration-Operations- Server- Server Control. Default: database_operations, event, reporting, scheduler, smartstate, ems_operations, ems_inventory, user_interface, web_services
session_store	Where to store the session information for all web requests. The possible values are sql, memory, or cache. SQL stores the session information in the database regardless of the type of database server. Memory stores all the session information in memory of the server process. Cache stores the information in a memcache server. Default: cache
startup_timeout	The amount of time in seconds that the server will wait and prevent logins during server startup before assuming the server has timed out starting and will redirect the user to the log page after login. Default: 300
vnc_port	If using VNC for remote console, the port used by VNC. Default: 5800
zone	Set the Zone for this appliance belongs. This is the same as Zone in Configuration-Operations- Server-Basic Information. Default : default
:worker_monitor	Starts and monitors the workers. Parameters specified here will override those set in the workers:default section.
poll	How often the worker monitor checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 15.seconds
miq_server_time_threshold	How much time to give the server to heartbeat before worker monitor starts to take action against non-responding server. Default: 2.minutes
nice_delta	Tells the worker monitor what Unix "nice" value to assign the workers when starting. A lower number is less nice to other processes. Default: 1
sync_interval	Time interval to sync active roles and configuration for all workers. Default: 30.minutes
wait_for_started_timeout	How long to wait for a started worker to heartbeat before considering the worker timed out. Default: 10.minutes
kill_algorithm	

Parameters	Description
name	Criteria used to start killing workers. Default: used_swap_percent_gt_value
value	Value of the criteria used. Default: 80
start_algorithm	
name	After server startup, criteria that must be met to decide if the Infrastructure Automation Server can start a new worker. Default: used_swap_percent_lt_value
value	Value of criteria used. Default: 60

Table: session

Parameters	Description
interval	Set the time interval in seconds for checking inactive sessions in appliance console. Default: 60
timeout	Set the time period in seconds in which inactive console sessions are deleted. This is the same as Session Timeout in Configuration-Operations-Server-Authentication in the appliance console. Default: 3600
memcache_server	If you choose memory for session_store, you need to specify the memcache_server to retrieve the session information from. Default: 127.0.1.1:11211
memcache_server_opts	Options to send to memcache server. : blank
show_login_info	Specify whether or not you want to see login info on start page. Default: true

Table: smartproxy_deploy

Parameters	Description
queue_timeout	Timeout for host smartproxy deploy job. Default: 30.minutes

Table: smtp

Parameters	Description
host	Specify the hostname of the smtp mail server. This is the same as Host in Configuration-Operations-Server-Outgoing SMTP E-mail Server. Default: localhost
port	Specify the port of the smtp mail server. This is the same as Port in Configuration-Operations-Server-Outgoing SMTP E-mail Server. Default: “25”
domain	Specify the domain of the smtp mail server. This is the same as Domain in Configuration-Operations-Server-Outgoing SMTP E-mail Server. Default: mydomain.com
authentication	Specify the type of authentication of the smtp mail server. This is the same as Authentication in Configuration-Operations-Server-Outgoing SMTP E-mail Server. Default: login
user_name	Specify the username required for login to the smtp mail server. This is the same as User Name in Configuration-Operations-Server-Outgoing SMTP E-mail Server. Default: evmadmin
password	Specify the encrypted password for the user_name account. This is the same as Password in Configuration-Operations-Server-Outgoing SMTP E-mail Server. Default: blank
from	Set the address that you want to send e-mails from. This is the same as From E-mail Address in Configuration-Operations-Server-Outgoing SMTP E-mail Server. Default: cfadmin@cfserver.com

Table: snapshots

Parameters	Description
create_free_percent	Ensures the % of free space available on the main datastore (datastore where vmx file is located) can support the % growth of the snapshot. The default is to require space for 100% of the provisioned size of all disks that are taking part in the snapshot. A value of 0 means do not check for space before creating the snapshot. Default: 100
remove_free_percent	Ensures the % of free space available on the main datastore (datastore where vmx file is located) has the % free space available to support the snapshot deletion process. Note that the deletion process consists of first composing a new snapshot then removing it once the original snapshot to be deleted has been collapsed in the VM. The default is to require 100% of the size of all disks to complete this process. A value of 0 means do not check for space before removing the snapshot. Default: 100

Table: webservices

Parameters	Description
contactwith	Set to ipaddress to contact miqhost using the IP address. Set to hostname to contact miqhost by its hostname. Set to resolved_ipaddress to take the hostname and resolve it to an IP address. Default: ipaddress
mode	Set to invoke to use webservices. Set to disable to turn off webservices. This is the same as Mode in Configuration-Operations-Server-Web Services in the appliance console. Default: invoke
nameresolution	If set to true, the hostname will be resolved to an IP address and saved with the host information in the VMDB. Default: false
security	If Web Services are enabled, you can set this to ws-security. This is the same as Security in Configuration-Operations- Server-Web Services in the appliance console. Note: This is not currently supported. Default: none
timeout	Specify the web service timeout in seconds. Default: 120
password	Specify the encrypted password for the user_name account. This is the same as Password in Configuration-Operations-Server-Outgoing SMTP E-mail Server. Default: blank

Parameters	Description
use_vim_broker	Controls if the vim_broker is used to communicate with VMware infrastructure. Default: true

Table: workers

Parameters	Description
worker_base	
defaults	If the following parameters are NOT explicitly defined for a specific worker, then these values will be used.
count	Number of this type of worker. Default: 1
gc_interval	How often to do garbage collection for this worker. Default: 15.minutes
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 3.seconds
poll_method	If set to normal, the worker checks for work the number of seconds set in the poll parameter. If set to escalate, the worker will increase the time between checks when there is no work to be done. Default: normal
poll_escalate_max	The maximum number of time to wait between checks for work. Poll_method must be set to escalate for this option to be used. Default: 30.seconds
heartbeat_freq	How often to “heartbeat” the worker. Default: 60.seconds
heartbeat_method	Set which way to dispatch work. Possible values are sql or drb. Default: drb
heartbeat_timeout	How long to wait until the worker heartbeat is considered timed out. Default: 2.minutes
parent_time_threshold	How long to allow the parent to go without heartbeating before considering the “parent’ not responding. For workers, the worker monitor is the parent. For Worker monitor, the Server is the parent. Default: 3.minutes
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 150.megabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 10
restart_interval	How long to let a worker remain up before asking it to restart. All queue based workers are set to 2.hours and every other worker does not get restarted by a 0.hours value. Default: 0.hours
starting_timeout	How long to wait before checking a worker’s heartbeat when it is starting up to mark it as not responding, similar to a grace period before you begin monitoring it. Default: 10.minutes
event_catcher	Associated with Event Monitor Server Role. Captures ems events and queues them up for the event_handler to process. Parameters specified here will override those set in the worker_base:default section.
ems_event_page_size	Internal system setting which sets the maximum page size for the event collector history. This should not be modified. Default: 100
ems_event_thread_shutdown_timeout	Internal system setting which determines how long the event catcher at shutdown will wait for the event monitor thread to stop. This should not be modified. Default: 10.seconds
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 2.gigabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 1
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 1.seconds
event_catcher_redhat	Contains settings that supersede the event_catcher for event_catcher_redhat.
event_catcher_vmware	Contains settings that supersede the event_catcher for event_catcher_vmware.
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 1.seconds
event_catcher_openstack	Contains settings that supersede the event_catcher for event_catcher_openstack.
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 15.seconds
topics	List of AMQP topics that should be monitored by Infrastructure Automation when gathering events from OpenStack.
duration	Qpid Specific. Length of time (in seconds) the receiver should wait for a message from the Qpid broker before timing out. Default: 10.seconds
capacity	Qpid Specific. The total number of messages that can be held locally by the Qpid client before it needs to fetch more messages from the broker. Default: 50.seconds

Parameters	Description
amqp_port	Port used for AMQP. Default: 5672
schedule_worker	Settings for Scheduler Server Role and any other work that runs on a schedule. Parameters specified here will override those set in the worker_base:default section.
db_diagnostics_interval	How frequently to collect database diagnostics statistics. Default: 30.minutes
job_proxy_dispatcher_interval	How often to check for available SmartProxies for SmartState Analysis jobs. Default: 15.seconds
job_proxy_dispatcher_stale_message_check_interval	How often to check for the dispatch message in the queue Default: 60.seconds
job_proxy_dispatcher_stale_message_timeout	Kill a message if this value is reached. Default: 2.minutes
job_timeout_interval	How often to check to see if a job has timed out. Default: 60.seconds
license_check_interval	How often to check for valid license. Default: 1.days
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 150.megabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 3
performance_collection_interval	Controls how often the schedule worker will put performance collection request on the queue to be picked up by the collection worker. Default: 3.minutes
performance_collection_start_delay	How long after Infrastructure Automation Server has started before starting capacity and utilization collection, if collection needs to be done. Default: 5.minutes
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 15.seconds
server_logs_stats_interval	How often to log the Infrastructure Automation Server statistics. Default: 5.minutes
server_stats_interval	How often to collect the Infrastructure Automation Server statistics. Default: 60.seconds
session_timeout_interval	How often to check to see if a UI (appliance console) session has timed out. Default: 30.seconds
storage_file_collection_interval	How often to perform file inventory of storage locations. Default: 1.days
storage_file_collection_time_utc	What time to perform file inventory of storage locations. Default: “06:00”
vdi_refresh_interval	How often to refresh vdi inventory. Default: 20.minutes
vm_retired_interval	How often to check for virtual machines that should be retired. Default: 10.minutes
vm_scan_interval	How often to check virtual machines to see if scan needs to be done. Default: 10.minutes
smis_refresh_worker	Settings for Storage Inventory Server Role and any other work that runs on a schedule. Parameters specified here will override those set in the worker_base:default section
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 15.seconds
connection_pool_size	Maximum number of database connections allowed per process. Default: 5
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 1.gigabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 3
smis_update_period	How frequently to update smis information. Default: 1.hours
status_update_period	How frequently to update smis status. Default: 5.minutes

Parameters	Description
stats_update_period	How frequently to update smis statistics. Default: 10.minutes
vim_broker_worker	Launched for any of these roles: Capacity & Utilization Collector, SmartProxy, SmartState Analysis, Management System Operations, Management System Inventory. Also launched if the use_vim_broker setting is on. Provides connection pooling, caching of data to and from the VMware infrastructure. Parameters specified here will override those set in the workers:default section.
heartbeat_freq	How often to heartbeat the worker. Default: 15.seconds
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 1.gigabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 3
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 1.seconds
reconnect_retry_interval	Period after which connection is retried. Default: 5.minutes
vim_broker_status_interval	Internal system setting which configures how much time to wait after receiving event updates before checking for more updates. Default: 0.seconds
wait_for_started_timeout	Time between the worker’s preload and startup time before considering the worker timed out. Default: 10.minutes
ui_worker:	Settings for User Interface Server Role. Parameters specified here will override those set in the worker_base:default section.
connection_pool_size	Maximum number of database connections allowed per process. Default: 5
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 1.gigabytes
nice_delta: 1	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 1
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 60.seconds
web_service_worker	Settings for Web Services Server Role. Parameters specified here will override those set in the worker_base:default section.
connection_pool_size	Maximum number of database connections allowed per process. Default: 5
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 1.gigabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 1
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 60.seconds
queue_worker_base	Base class of all queue workers that work off of the queue..
defaults	If the following parameters are NOT explicitly defined for a queue worker, then these values will be used.
cpu_usage_threshold	How much cpu to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 100.percent
queue_timeout	How long a queue message can be worked on before it is considered timed out. Default: 10.minutes
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 400.megabytes
restart_interval	Queue workers restart interval. Default: 2.hours
poll_method	If set to normal, the worker checks for work the number of seconds set in the poll parameter. If set to escalate, the worker will increase the time between checks when there is no work to be done. Default: normal
generic_worker	Performs work that is not classified as any specific type of work. Processes all normal priority or non-specific queue items. Parameters specified here will override those set in the queue_worker_base:default section
count	Number of this type of worker. Default: 4
ems_refresh_worker	Performs all ems (management system) refreshes to keep the vmdb in sync with the state of the components of the virtual infrastructure in the various management systems. Parameters specified here will override those set in the queue_worker_base:default section
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 10.seconds

Parameters	Description
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 2.gigabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 7
restart_interval	Queue workers restart interval. Default: 2.hours
queue_timeout	How long a message can be worked on before it is considered timed out. Default: 120.minutes
event_handler	Associated with Event Monitor Server Role. Handles all events caught by the event catcher worker. Parameters specified here will override those set in the workers:default section. Parameters specified here will override those set in the queue_worker_base:default section
cpu_usage_threshold	How much cpu to allow the worker to grow to before gracefully requesting it to exit and restart. The value of 0 means that this worker will never be killed due to CPU usage. Default: 0.percent
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 7
perf_collector_worker	Connects to VC/ESX to collect the raw performance data. Same as the Capacity & Utilization Data Collector Server Role. Parameters specified here will override those set in the queue_worker_base:default section count. Number of this type of worker. Default: 2
count	Number of this type of worker. Default: 2
poll_method	If set to normal, the worker checks for work the number of seconds set in the poll parameter. If set to escalate, the worker will increase the time between checks when there is no work to be done. Default: escalate
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 3
perf_processor_worker	Processes the raw performance data into a reportable format. Same as the Capacity & Utilization Data Processor Server Role. Parameters specified here will override those set in the queue_worker_base:default section
count	Number of this type of worker. Default: 2
poll_method	If set to normal, the worker checks for work the number of seconds set in the poll parameter. If set to escalate, the worker will increase the time between checks when there is no work to be done. Default: escalate
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 400.megabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 7
priority_worker	Performs all high priority queue items including many tasks on behalf of the UI. UI requests are normally executed by a priority worker so they will not to block the UI. Parameters specified here will override those set in the queue_worker_base:default section
count	Number of this type of worker. Default: 2
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 200.megabytes
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 1
poll	How often the workers checks for work. This value only is only used when the worker has no more work to do from the queue. It will wait for an amount of time determined by the poll value and poll method. Therefore, if there is constant work on the queue, the worker will not wait in between messages. Default: 1.seconds
reporting_worker	Compiles reports. Settings for Reporting Server Role. Parameters specified here will override those set in the queue_worker_base:default section
count	Number of this type of worker. Default: 2
nice_delta	Tells the worker monitor what Unix “nice” value to assign the workers when starting. A lower number is less nice to other processes. Default: 7
smart_proxy_worker	Performs the embedded scanning of virtual machines. Settings for SmartProxy Server Role. Parameters specified here will override those set in the queue_worker_base:default section
count	Number of this type of worker. Default: 3
memory_threshold	How much memory to allow the worker to grow to before gracefully requesting it to exit and restart. Default: 600.megabytes
queue_timeout	How long a queue message can be worked on before it is considered timed out. Default: 20.minutes
restart_interval	Queue workers restart interval. Default: 2.hours

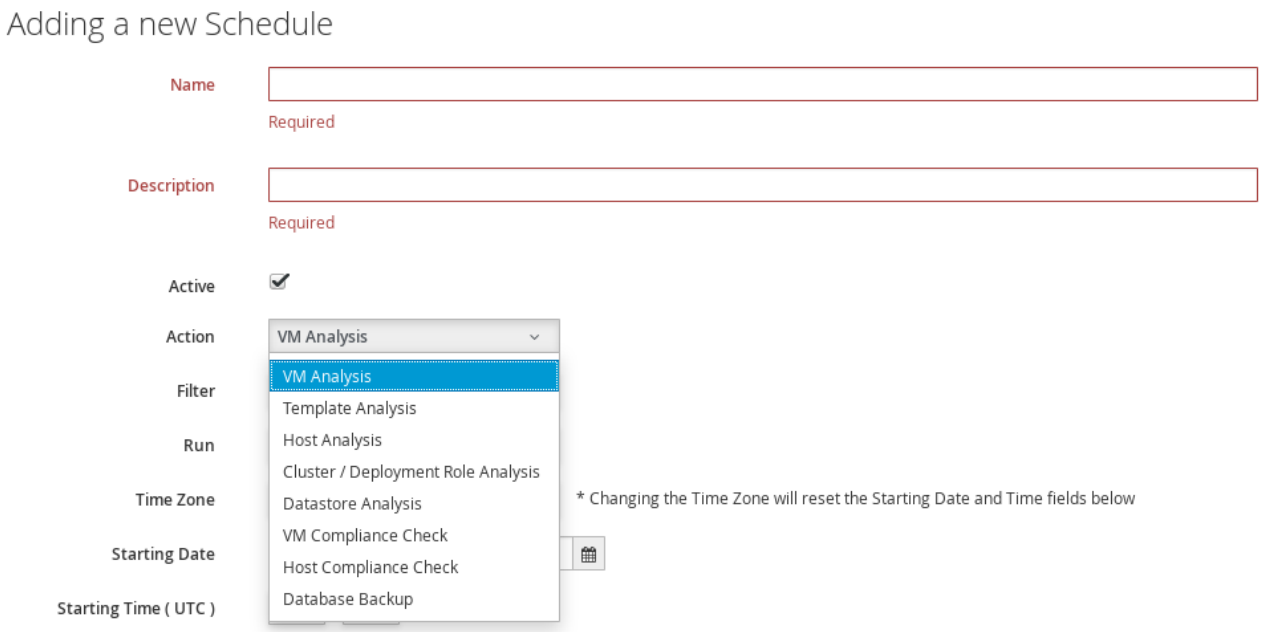
Schedules

Scheduling SmartState Analyses

From the **Schedules** area in **Settings**, you can schedule the analyses of virtual machines, hosts, clusters, and datastores to keep the information current. Depending on which resource you want to analyze, you can filter which ones to analyze. You may also specify only one virtual machine or perform an analysis on all virtual machines. In addition, you can schedule compliance checks, and database backups.

To schedule a SmartState Analysis or Compliance Check:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Schedules**.
3. Click (**Configuration**), and (**Add a new Schedule**).
4. In the **Basic Information** area, type in a **Name** and **Description** for the schedule.
5. Select **Active** to enable this scan.
6. From the **Action** list, select the type of analysis to schedule. Based on the type of analysis you choose, you are presented with one of the following group boxes:



- **VM Analysis**: Displays **VM Selection** where you can choose to analyze **All VMs**, **All VMs for Provider**, **All VMs for Cluster**, **All VMs for Host**, **A single VM**, or **Global Filters**.
- **Template Analysis**: Displays **Template Selection** where you can choose to analyze **All Templates**, **All Templates for Provider**, **All Templates for Cluster**, **All Templates for Host**, **A single Template**, or **Global Filters**.
- **Host Analysis**: Displays **Host Selection** where you can choose to analyze **All Hosts**, **All Hosts for Provider**, **A single Host**, or **Global Filters**.

  Note:

  You can only schedule host analyses for connected virtual machines, not repository virtual machines that were discovered through that host. Since repository virtual machines do not retain a relationship with the host that discovered them, there is no current way to scan them through the scheduling feature. The host is shown because it may have connected virtual machines in the future when the schedule is set to run.
- **Cluster / Deployment Role Analysis**: Displays **Cluster Selection** where you can choose to analyze **All Clusters**, **All Clusters for Provider**, or **A single Cluster**.
- **Datastore Analysis**: Displays **Datastore Selection** where you can choose to analyze **All Datastores**, **All Datastores for Host**, **All Datastores for Provider**, **A single Datastore**, or **Global Filters**.
- **VM Compliance Check**: Displays **VM Selection** where you can choose to analyze **All VMs**, **All VMs for Provider**, **All VMs for Cluster**, **All VMs for Host**, **A single VM**, or **Global Filters**.
- **Host Compliance Check**: Displays **Host Selection** where you can choose to analyze **All Hosts**, **All Hosts for Provider**, **All Hosts for Cluster**, **A single Host**, or **Global Filters**.
- **Database Backup**: Under **Type**, displays **Network File System**, **Samba**, **Amazon AWS S3** and **OpenStack Swift**..

7. By applying **Global Filters** within any of the above items, you can designate which virtual machines or hosts to analyze.

8. In **Run**, set the frequency of the analysis to run. There are further options based on which **Run** option you choose.
    - Click **Once** to have the analysis run only one time.
    - Click **Daily** to run the analysis on a daily basis. You will be prompted to select the number of days between each analysis.
    - Click **Hourly** to run the analysis hourly. You will be prompted to select the number of hours between each analysis.
9. Select a **Time Zone**.

    Note:

    If you change the **Time Zone**, you will need to reset the stating date and time.
10. Type or select a date to begin the schedule in **Starting Date**.
11. Select a **Starting Time** based on a 24 hour clock in the selected Time Zone.
12. Click **Add**.

Modifying a Schedule

To modify a schedule:

1. Browse to **Settings > Application Settings**.
2. Click on the **Settings** accordion, then click **Schedules**.
3. Click the schedule that you want to edit.
4. Click (**Configuration**), and then click (**Edit this Schedule**).
5. Make the required changes.
6. Click **Save**.

Importing and Exporting Schedules

Infrastructure Automation provides the ability to import and export schedules using the **`import_export_schedules.rb`** script:

```
./var/www/miq/vmdb/tools/import_export_schedules.rb -h
```

Option	Description
-u, –user=<s>	userid (default: admin)
-d, –output-dir=<s>	Output directory (default: ./)
-s, –schedule=<s>	Schedule name or id
-o, –operation=<s>	Export or import (default: export)
-y, –import-yaml=<filename/uri>	Imported yaml
-h, –help	Show this message

Usage Example.

Import:

```
./var/www/miq/vmdb/tools/import_export_schedules.rb -o import -y ./filename.yaml
```

Export:

```
./var/www/miq/vmdb/tools/import_export_schedules.rb -s 158
```

Note that the **`import_export_schedules.rb`** script works for all types of schedules, including:

- Report
- Policy
- Alert
- SmartState Analysis
- Automation Task
- Service Template

Access Control

Browse to **Settings > Application Settings**. Click on the **Access Control** accordion to see a hierarchy of the configurable items for users, groups, roles, and tenants. You can add and modify users, groups, account roles, tenants, and projects.

Creating a Tenant

Tenants are used in Infrastructure Automation for controlling visibility of objects.

To create a tenant:

1. Browse to **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Tenants**.
3. Click on the top-level **Tenant**, click (**Configuration**), and (**Add child Tenant to this Tenant**) to create a tenant.
4. Enter a name for the tenant in the **Name** field.
5. Enter a description for the tenant in the **Description** field.
6. Click **Add**.

Creating a Project

A tenant can also contain a self-contained child tenant known as a 'project'. A project cannot have a child tenant, but it is useful for allocating resources to a small group or team within a larger organization.

To create a project:

1. Browse to **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Tenants**.
3. Click on the **Tenant** where you want to add a **Project**, click (**Configuration**), and (**Add Project to this Tenant**) to create a project.
4. Enter a name for the project in the **Name** field.
5. Enter a description for the project in the **Description** field.
6. Click **Add**.

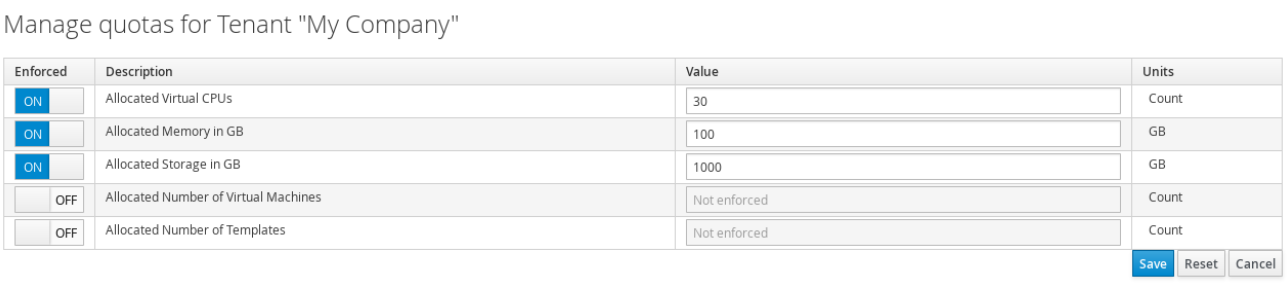
Managing Tenant and Project Quotas

Use the following procedure to allocate or edit quotas for tenants and projects.

Note:

Quota is allocated based on the user's current group. If the user belongs to multiple groups, you must change to the desired group before allocating or editing group quota. See Groups for more details.

1. Browse to **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Tenants**.
3. Click on the **Tenant** or **Project** where you want to add a quota, click (**Configuration**), and (**Manage quotas for the Selected Item**) to create a quota.
4. In the list of pre-built quotas, switch **Enforced** next to the quota item you want to enable to `Yes`.
5. In the **Value** field, enter the constraints you want to apply to the quota.



6. Click **Save**.

Tagging Tenants and Projects

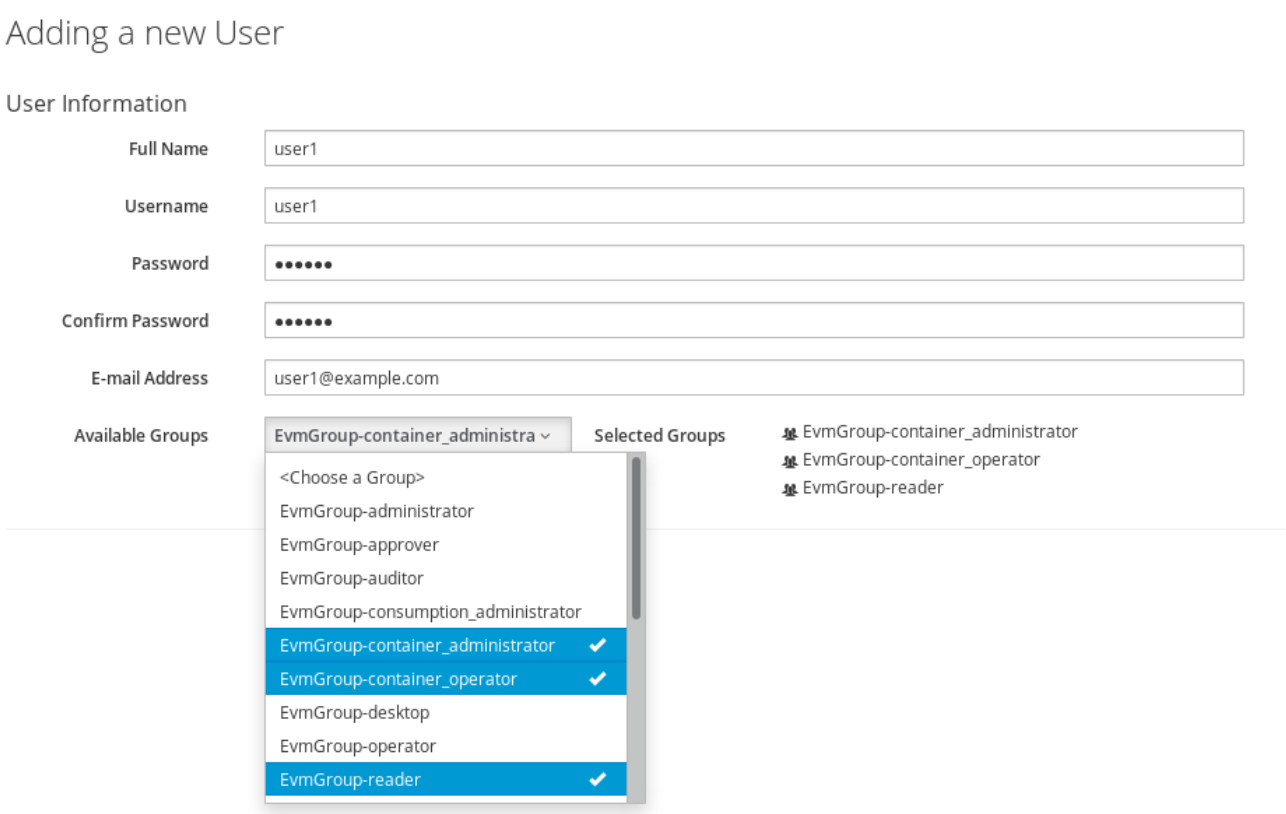
To tag tenants and projects:

1. Browse to **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Tenants**.
3. Select the tenant or project, then click **Policy**, and select **Edit My Company Tags for this Tenant**.
4. In **Tag Assignment**, click **Select a customer tag to assign**, and select a tag from the list. In the next column, set a corresponding value.
5. Click **Save**.

Creating a User

To create a user:

1. Browse to **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Users**.
3. Click (**Configuration**), and (**Add a new User**) to create a user.
4. Enter a **Full Name**, **Username**, **Password** with confirmation, and **Email Address** for the user. An email address is required; omitting the user email address can result in unsuccessful provisioning requests.



- If you are using LDAP, but did not enable **Get User Groups from LDAP** in your server's **Authentication** tab, you will need to define a user. The UserID must match exactly the user's name as defined in your directory service. Use all lowercase characters to ensure the user can be found in the VMDB. - When the user logs in, they use their LDAP password.

5. Select one or more groups from **Available Groups**.
6. Click **Add**.

Deleting a User

For security reasons, delete any user that no longer needs access to the information or functions of the server.

To delete a user:

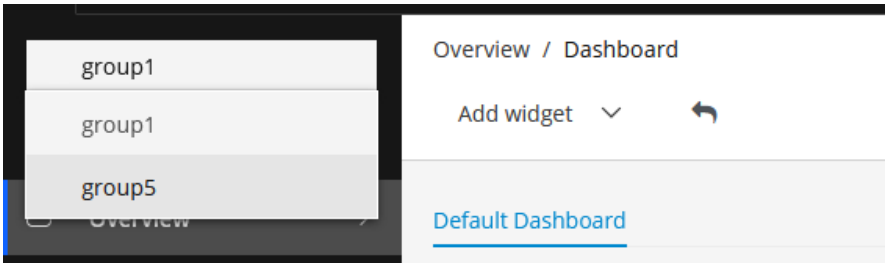
1. Browse to **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Users**.
3. Select the user you want to delete.
4. Click (**Configuration**), and (**Delete selected Users**) to delete a user.

Groups

User groups create filters and assign roles to users. You can either create your own groups, or leverage your LDAP directory service to assign groups of users to account roles. For a list of what each pre-defined account role can do, see Roles.

A user can exist in multiple groups. However, a group can only be assigned one account role.

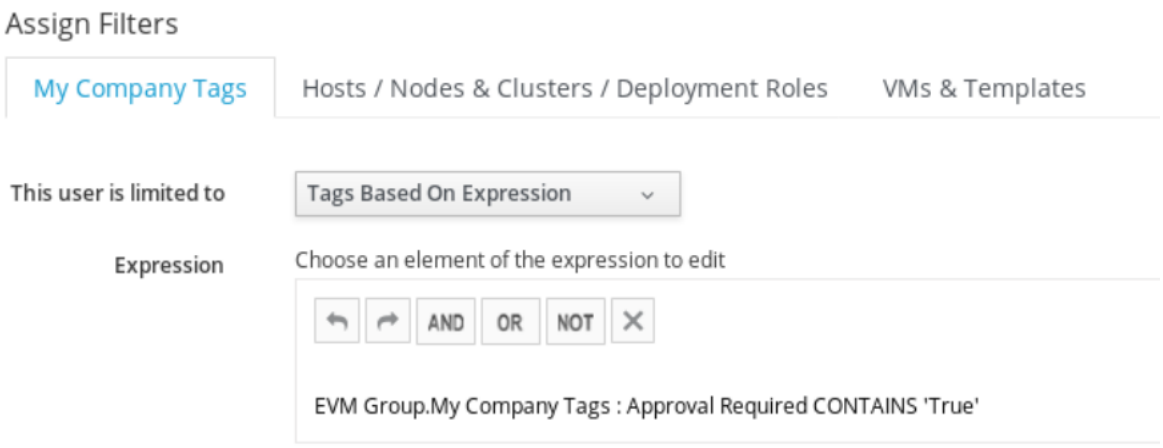
If you belong to multiple groups, you can change the current group. The Current Group affects quota and other settings for the user, including permissions for controlling the tasks that you can complete and the data that you can view. To change the group, click the group name in the upper left of the screen, and select your preferred group from the listed groups.



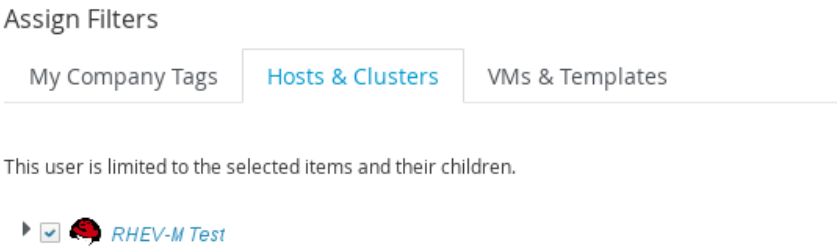
Creating a Group

To create a user group:

1. Browse to **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Groups**.
3. Click (**Configuration**), and (**Add a new Group**) to create a group.
4. Enter a name for the group in the **Name** field. To ensure compatibility with tags, use underscores in place of spaces. For example, Infrastructure Automation-`test_group`.
5. Select a **Role** to map to this group. For a description of each Infrastructure Automation role, see Account Roles and Descriptions.
6. Select the **Project/Tenant** this group must belong to.
7. Limit what users in this group can view by selecting filters in the **Assign Filters** area.
    1. Click the **<My Company> Tags** tab to select the tags that users in this group can access. Resources with the selected tags attached can be accessed by this group. Select tags using one of the options in the **This user is limited to** list:
        - Select **Specific Tags**, then check the boxes for the tags that you want to limit this user to. The items that have changed will show in blue italicized font.
        - Select **Tags Based On Expression**, then create tags based on an expression using AND, OR, or NOT. This allows you to further limit the resources accessible to a user: for example, to specify a combination of tags that must exist on a resource.



2. Click the **Host & Clusters** tab.
    1. Check the boxes for the host and clusters that you want to limit this user to. The items that have changed will show in blue

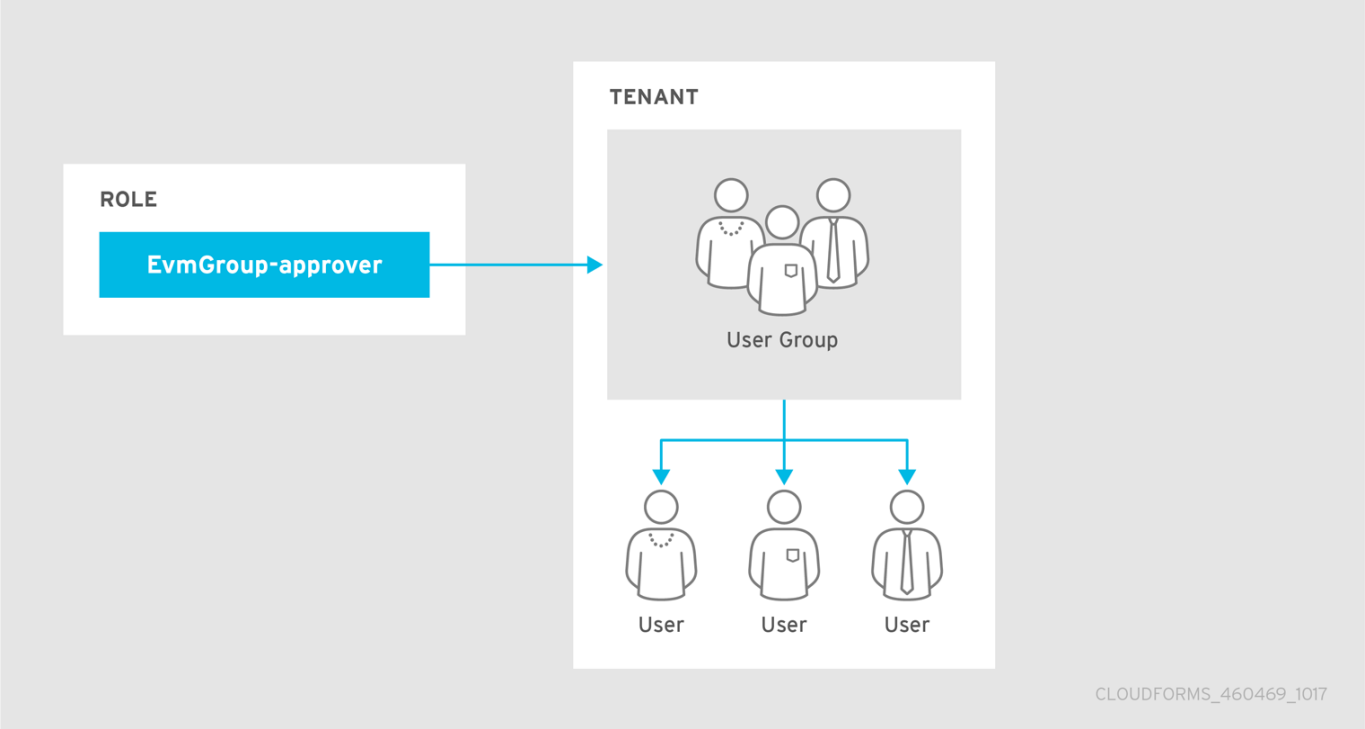


    italicized font.
3. Click the **VMs & Templates** tab. This shows folders that you have created in your virtual infrastructure.
    1. Check the boxes for the folders that you want to limit this user to. The items that have changed will show in blue italicized font.
8. Click **Add**.

After creating a group, assign one or more users to the group by editing a user.

Roles

When you create a group, you must specify a role to give the group rights to resources in the console. The group's role determines the scope of access for the users that are members of the group.



Infrastructure Automation provides a default group of roles, but you can also create your own, or copy and edit the default groups.

Note:

If you have enabled **Get Role from LDAP** under **LDAP Settings**, then the role is determined by the LDAP user's group membership in the directory service.

To view details of a role and its level of access:

1. Browse to **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Roles**.
3. Click on a role from the list to display role information and the product features the role can access (marked by a checkmark). You can expand the categories under **Product Features** to see further detail.

The table below shows a summary of the functions available to each role.

Account Roles and Descriptions

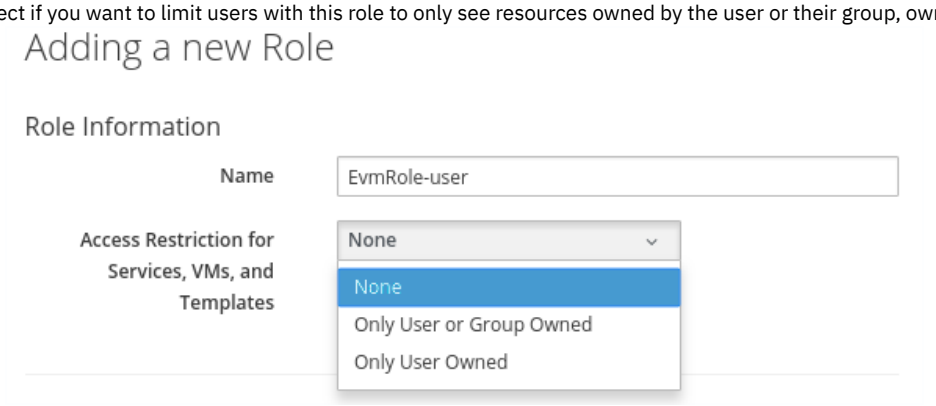
Role	Description
Administrator	Administrator of the virtual infrastructure. Can access all infrastructure functionality. Cannot change server configuration.
Approver	Approver of processes, but not operations. Can view items in the virtual infrastructure, view all aspects of policies and assign policies to policy profiles. Cannot perform actions on infrastructure items.
Auditor	Able to see virtual infrastructure for auditing purposes. Can view all infrastructure items. Cannot perform actions on them.
Container Administrator	Administrator with capabilities to configure, view and execute tasks on all containers and related underlying infrastructure. Has access to Nodes, Pods and Projects dashboards.
Container Operator	This role can view and execute tasks related to containers and related underlying infrastructure. The Container Operator has access to locked versions of the same dashboards as the Container Administrator.
Desktop	Access to VDI pages.
Operator	Performs operations of virtual infrastructure. Can view and perform all functions on virtual infrastructure items including starting and stopping virtual machines. Cannot assign policy, but can view policy simulation from Virtual Machine page.
Security	Enforces security for the virtual environment. Can assign policies to policy profiles, control user accounts, and view all parts of virtual infrastructure. Cannot create policies or perform actions on virtual infrastructure.
Super Administrator	Administrator of Infrastructure Automation and the virtual infrastructure. Can access all functionality and configuration areas.
Support	Access to features required by a support department such as diagnostics (logs). Can view all infrastructure items and logs. Cannot perform actions on them.
Tenant Administrator	Configures settings applicable to a Tenant. Sets Branding, maps groups/roles, configures LDAP credentials, and configures dashboard settings.

Role	Description
Tenant Quota Administrator	Configures quota limits for the tenant, applying usage constraints for CPU, Memory, Storage, Maximum number of VMs, and Maximum number of Templates.
User	User of the virtual infrastructure. Can view all virtual infrastructure items. Cannot perform actions on them.
User Limited Self Service	Limited User of virtual machines. Can make provision requests. Can access some functions on the virtual machine that the user owns including changing power state.
User Self Service	User of virtual machines. Can make provision requests. Can access some functions on the virtual machine that the user owns and that the user's LDAP groups own including changing power state.
Vm User	User of virtual machines. Can access all functions on the virtual machine including changing power state and viewing its console. Cannot assign policy, but can view policy simulation from virtual machine page.

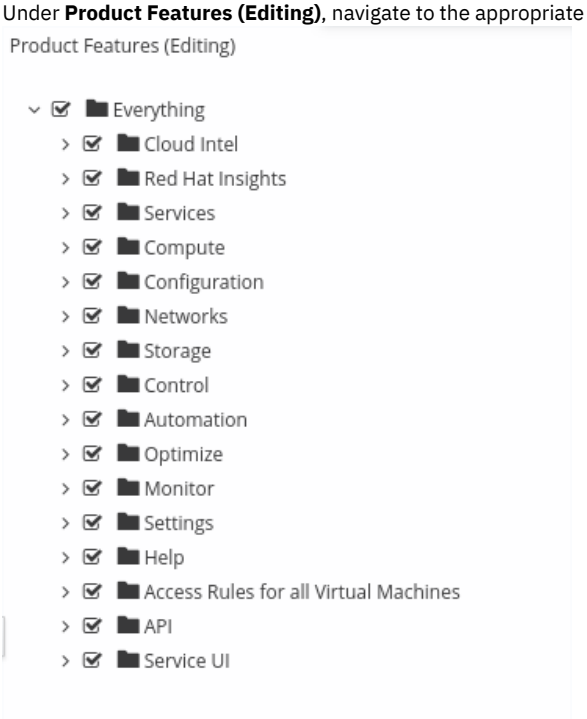
Creating a Role

To create a role:

1. Click **Settings > Application Settings**.
2. Click on the **Access Control** accordion, then click **Roles**.
3. Click **Configuration**, and **Add a new Role**. Alternatively, you can copy an existing role to a new role by clicking **Copy the selected Role to a new Role**.
4. In the **Role Information** area, type a name for the new role. For **Access Restriction for Catalog Items, Orchestration Stacks, Key Pairs, Services, VMs, and Templates**, select if you want to limit users with this role to only see resources owned by the user or their group, owned by the user, or all resources (**None**):



5. Under **Product Features (Editing)**, navigate to the appropriate feature and enable or disable it:



6. Click **Add**.

Diagnostics

Browse to **Settings > Application Settings**. Click on the **Diagnostics** tab to see the status of the different Infrastructure Automation roles and workers for each server, view and collect logs, and gather data if there are any gaps in capacity and utilization information. The Diagnostics area is designed in a hierarchy.

- At the **region** level, you can see replication status, backup the VMDB, and run garbage collection on the VMDB.
- At the **zone** level, you can see Infrastructure Automation roles by servers and servers by roles. In addition, you can set log collection values for a specific zone, and collect gap data for capacity and utilization.
- At the **server** level, you can see the workers for each server, set log collection values for a specific server, and view current logs.

Region Diagnostics

Using the console, you can set the priority of server regional roles, review and reset replication, and create backups of your database either on demand or on a schedule.

Regions are used primarily to consolidate multiple VMDBs into one master VMDB for reporting while zones are used to define functional groups of servers. There can be only one region per VMDB, but multiple zones per region (or VMDB). Some server roles are aware of each other across Infrastructure Automation appliances at the region level. This means that redundancy and failover rules apply at the region level. You can also set priorities for the server roles that provide failover.

Server Role Priorities

If you have multiple servers in your environment with duplicate failover roles, then you can set the priority of the server role.

- Only server roles that support failover can be marked as primary. These roles only allow one server to be active at a time. These are **Notifier**, **Capacity & Utilization Coordinator**, **Event Monitor**, **Scheduler**, **Storage Inventory**, and **Provider Inventory**.
- All other server roles are additive. The more servers with that role in a zone the more work that can be performed.

There are three role priorities.

- **Primary**: There can only be one primary per zone or region per role. When an appliance is started, the system looks to see if any role is set to primary. If that is the case, the role is activated on that appliance and deactivated from the secondary. In the console, primary roles are shown in bold letters. The text turns red if the server goes down. You must actively set the primary priority.
- **Secondary**: This is the default priority. There can be multiple secondaries. When an appliance is started, if no primary is found in the zone, the first appliance to start takes the role. In the console, secondary roles are displayed normally with the word “secondary”.
- **Tertiary**: If all appliances with primary roles or secondary roles were down, one of the tertiary would be activated. The reason for tertiary is to ensure that if a server with crucial roles such as Provider Inventory or Event Monitor goes down, you have a way to associate those roles to different appliances by organizing the priorities. Tertiary roles simply show as active in the console.

Region Aware Server Roles

Role	More than one per Region	Can have Priority Set
Automation Engine	Y	N
Database Operations	Y	N
Notifier	N	Y
Reporting	Y	N
Scheduler	N	Y
User Interface	Y	N
Web Services	Y	N

Setting the Priority of a Failover Role

To set the priority of a failover role:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Depending on how you want to view your servers, click either the **Roles by Servers** tab or the **Servers by Roles** tab.
4. In the **Status** of **Roles for Servers** in **Zone Default Zone** area, click on the role that you want to set the priority for.
5. Click (**Configuration**), and (**Promote Server**) to make this the primary server for this role.
6. Click (**Configuration**), and (**Demote Server**) to demote the priority of this server for this role.

Zone Diagnostics

The console provides a way to see all the server roles that a server has been assigned and if these roles are running. This is especially helpful when you have multiple servers with different server roles. For each zone you can also set a central place for all logs to be collected, and collect capacity and utilization data that may be missing.

Viewing the Status of Server Roles

To view the status of server roles:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Depending on how you want to view your servers, click either **Roles by Servers** or the **Servers by Roles**.

Zone Aware Server Roles

Role	More than one per Region	Can have Priority Set
Automation Engine	Y	N
Capacity & Utilization Coordinator	N	Y
Capacity & Utilization Data Collector	Y	N
Capacity & Utilization Data Processor	Y	N
Database Operations	Y	N
Event Monitor	N	Y
Provider Inventory	N	Y
Provider Operations	Y	N
Notifier	N	Y
Reporting	Y	N
Scheduler	N	Y
SmartProxy	Y	N
SmartState Analysis	Y	N
User Interface	Y	N
Web Services	Y	N

Removing an Inactive Server

To remove an inactive server:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Click on the name of the server in the tree view.
4. Click (**Delete Server**). This button is available only if the server is inactive.

Zone Log Collections

If you have multiple servers reporting to one central VMDB, then you can collect the configuration files and logs from the console of any of the servers. While you can set this either at the zone or server level, settings at the server level supersede the ones at the zone level.

Log depot options include:

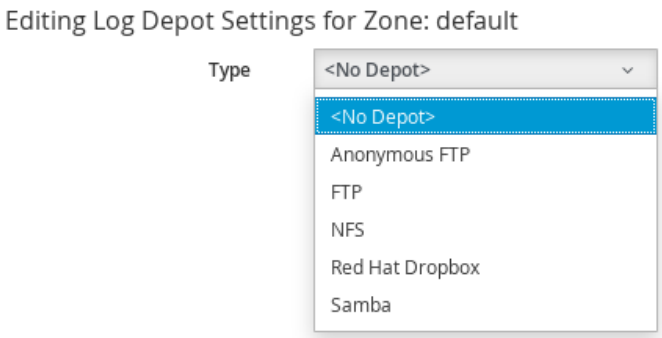
- Anonymous File Transfer Protocol (FTP)
- File Transfer Protocol (FTP)
- Network File System (NFS)
- Red Hat Dropbox
- Samba

See your network administrator if you need to set up one of these shares. You will also need a user that has write access to that location.

Setting the Location of the Log Depot

To set the location of the log depot:

1. Browse to **Settings > Application Settings**.
2. Click the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Click **Collect Logs**.
4. Click (**Edit**).
5. Select the **Type** of share.
6. Using the fully qualified domain name (**FQDN**) of the depot server, type in the appropriate settings for the **URI**.



7. If required, enter your user **ID** and **password** then click **Validate** to confirm the settings.
8. Click **Save**.

Collecting and Downloading Logs from All Servers in a Zone

To collect and download logs from all servers in a zone:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Click the **Collect Logs** tab.
4. Click (**Collect all logs**). All files in the logs directory as well as configuration files are collected from the selected zone.
5. Click **OK**. The status of the log retrieval shows in the Infrastructure Automation console.

Capacity and Utilization Repair

Under certain circumstances, it is possible that Infrastructure Automation is not able to collect capacity and utilization data. This could be due to password expiration, a change in rights to the cloud provider and this change didn't provide enough granularity to the Infrastructure Automation service account, or network connectivity. The gap data is collected directly by extracting the monthly performance data. Gap collection need to be completed for each zone individually. Therefore, the procedure below need to be repeated for each zone.

Repairing Capacity and Utilization Data

To repair capacity and utilization data:

1. Log in to a Infrastructure Automation appliance located in the zone for which you want to gather the data.
2. Browse to **Settings > Application Settings**.
3. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
4. Click **C & U Gap Collection**.
    1. Select the appropriate **Timezone**.

       Note:

       Do not select more than one week unless instructed to do so by Red Hat Support.
    2. Select a **Start Date**.
    3. Select an **End Date**.

5. Click **Submit**.

After the gap collection has completed for this zone, repeat these same steps for the next zone. You can check for completion by going to the clusters page and checking for the capacity and utilization data for the time period specified.

Server Diagnostics

Under **Diagnostics** for a server, you can view the status of Infrastructure Automation workers running on the server, set log collection setting for only that server, and view the server's current Infrastructure Automation and audit logs.

Workers

The **Workers** tab enables you to see the status of and restart Infrastructure Automation workers.

You can see additional information on and restart the following items:

- **C & U Metrics Collectors** that collects capacity and utilization data.
- **C & U Metrics Processors**, which processes the collected capacity and utilization data.
- **Event Handlers** put events from the Event Monitor into the VMDB and starts Infrastructure Automation processes if needed base on that information.
- **Event Monitors** that communicate with the external cloud provider to deliver up to date event information.
- **Generic Workers** that perform long running and priority processes.
- **Priority Workers** that perform high priority, short processes.
- **Schedule Workers** that maintains any items that run on a schedule.
- **Session Broker** that maintains a single connection to the cloud providers .
- **Refresh Workers** that runs the refresh processes.
- **Reporting Workers** that generate reports.
- **SmartProxy Workers** that run SmartState Analyses on virtual machine.
- **User Interface Worker** that allows users access to the console.
- **Web Services Worker** that maintains Infrastructure Automation Web services.
- **VM Analysis Collectors** that run and process SmartState Analyses on virtual machines.

Reloading Worker Display

To reload worker display:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click the **Workers** tab.
5. Click (**Refresh Current Workers display**).

Restarting a Worker

To restart a worker:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click on the **Workers** tab.
5. Click on the worker you want to restart.
6. Click (**Configuration**), then (**Restart selected worker**).

7. Click **OK**.

Server and Audit Logs

Collecting Server Logs and Configuration Files

While you can designate a central location to collect logs for all servers in a specific zone, you can override those values for a specific server. To do this, designate a log depot location to store the files.

Log depot options include:

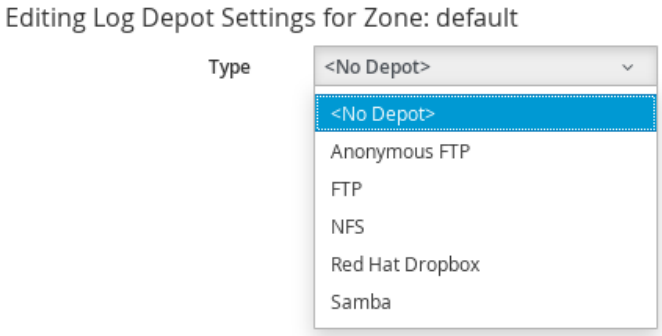
- Anonymous File Transfer Protocol (FTP)
- File Transfer Protocol (FTP)
- Network File System (NFS)
- Red Hat Dropbox
- Samba

See your network administrator to set up one of these shares. You also need a user that has write access to that location. Settings at the server level supersede the ones at the zone level.

Setting the Location of the Log Depot for a Specific Server

To set the location of the log depot for a specific server

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to collect logs for.
4. Click on the **Collect Logs** tab.
5. Click (**Edit Log Depot Settings for the selected Server**).
6. Select the **Type** of share.



7. Using the fully qualified domain name (**FQDN**) of the depot server, type in the appropriate settings for the **URI**.
8. Enter your user ID and password, then click **Validate** to confirm the settings.
9. Click **Save**.

Collecting the Current Log Set of a Server

To Collect the Current Log Set of a Server

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to collect logs for.
4. Click on the **Collect Logs** tab.
5. Click (**Collect**), then click (**Collect current logs**). All current log files in as well as configuration files are collected.

6. Click **OK**. The status of the log retrieval shows in the Infrastructure Automation console.

Collecting All Log Sets from a Server

To Collect All Log Sets from a Server

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to collect logs for.
4. Click **Collect Logs**.
5. Click (**Collect**), then click (**Collect all logs**). All files in the logs directory as well as configuration files are collected.
6. Click **OK**. The status of the log retrieval shows in the Infrastructure Automation console.

Viewing the Server, Audit, and Production Logs

The server and audit logs roll over daily. The previous logs are stored as zipped files in the `/var/www/miq/vmdb/log` folder. The current logs can be easily viewed and downloaded from the settings menu; select **Configuration**, then click on the **Diagnostics** accordion.

Use the server log to see all actions taken by the server including communication with the SmartProxy and tasks.

Viewing the Server Log

To view the server log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click **CFME Log**.

The Infrastructure Automation server automatically retrieves the last 1000 lines of the log.

Reloading the Server Log

To reload the server log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click **CFME Log**.
5. Click (Reload the Log Display).

Downloading the Server Log

To download the server log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click **CFME Log**.
5. Click (**Download the Entire EVM Log File**).

Note:

Use the **Audit Log** to see changes to the user interface and authentication.

Viewing the Audit Log

To view the audit log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click **Audit Log**.

The server automatically retrieves the last 1000 lines of the log.

Reloading the Audit Log

To reload the audit log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click **Audit Log**.
5. Click (**Reload the Audit Log Display**).

Downloading the Audit Log

To download the audit log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the Zone that you want to view.
3. Select the server that you want to view.
4. Click Audit Log.
5. Click (**Download the Entire Audit Log File**).

Viewing the Production Log

Use the production log to see all actions performed using the console.

To view the production log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click **Production Log**.

The Infrastructure Automation server automatically retrieves the last 1000 lines of the log.

Reloading the Production Log

To reload the production log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the Zone that you want to view.
3. Select the server that you want to view.
4. Click **Production Log**.
5. Click (**Reload the Product Log Display**).

Downloading the Production Log

To download the production log:

1. Browse to **Settings > Application Settings**.
2. Click on the **Diagnostics** accordion, then click the **Zone** that you want to view.
3. Select the server that you want to view.
4. Click **Production Log**.
5. Click (**Download the Production Log File**).

Database Operations

Viewing Information on the VMDB

The **Database** accordion displays a summary of VMDB information, a list of all tables and indexes, settings for the tables, active client connection, and database utilization.

To view information on the VMDB:

1. Browse to **Settings > Application Settings**.
2. Click the **Database** accordion.
3. Click **VMDB** in the tree view on the left.
4. Click the appropriate tab to view the desired information:
   - **Summary**: displays statistics about the database.
   - **Tables**: displays a clickable list of all the tables.
   - **Indexes**: displays a clickable list of all the indexes.
   - **Settings**: displays a list of all tables, their descriptions, and other valuable Information.
   - **Client Connections**: displays all current connections to the VMDB.
   - **Utilization**: displays usage charges for the disk and index nodes.

Database Regions and Replication

Regions are used to create a central database for reporting and charting. Do not use the database at the top level for operational tasks such as SmartState analysis or capacity and utilization data collection. Assign each server participating in the region a unique number during the regionalization process, then set your subordinate regions to replicate to the top region.

Important

All Infrastructure Automation databases in a multi-region deployment must use the same security key.

Creating a Region

In principle, a region is created when you set up your Infrastructure Automation environment on the first appliance for the region. However, you can also create a region on an appliance where a database has already been set up. This process involves dropping and rebuilding the existing database to accommodate the new region number, and takes several minutes to complete.

Warning

Performing this procedure destroys any existing data and cannot be undone. Back up the existing database before proceeding. By default, new appliances are assigned region 0. Do not use this number when creating a region because duplicate region numbers can compromise the integrity of the data in the database.

1. Log in to the appliance as the **root** user.
2. Enter **`appliance_console`**, and press **Enter**.
3. Press any key.
4. Select **`Stop EVM Server Processes`**.
5. Enter **`Y`** to confirm.
6. Choose **`Configure Application`**.
7. Enter a database region number that has not been used in your environment. Do not enter duplicate region numbers because this can corrupt the data.

8. Enter `Y` to confirm.
9. The menu reappears after all processes are complete.
10. Select `Start EVM Server Processes`.
11. Enter `Y` to confirm.
12. Select **Quit** to exit the advanced settings menu.

Configuring Database Replication and Centralized Administration

To configure database replication, you must configure one Infrastructure Automation instance to act as a global copy, and one or more other instances to act as remote copies. This database replication relationship can only be configured on Infrastructure Automation instances that are of the same version. Configuring database replication in this version of Infrastructure Automation automatically enables centralized administration, eliminating the need for further configuration. Centralized administration in Infrastructure Automation supports life cycle management operations that can be initiated from the global copy and processed and executed on the remote copy.

Important

- You must configure at least one remote copy before you can configure the global copy. Changes to the role of a Infrastructure Automation instance take several minutes to take effect.
- The region number must be unique on each Infrastructure Automation instance where replication is configured. See Creating a Region for instructions on how to create a region.
- Enabling the `EvmRole-user_global_region` role will only provide access to specific operations that can be performed on a global region.

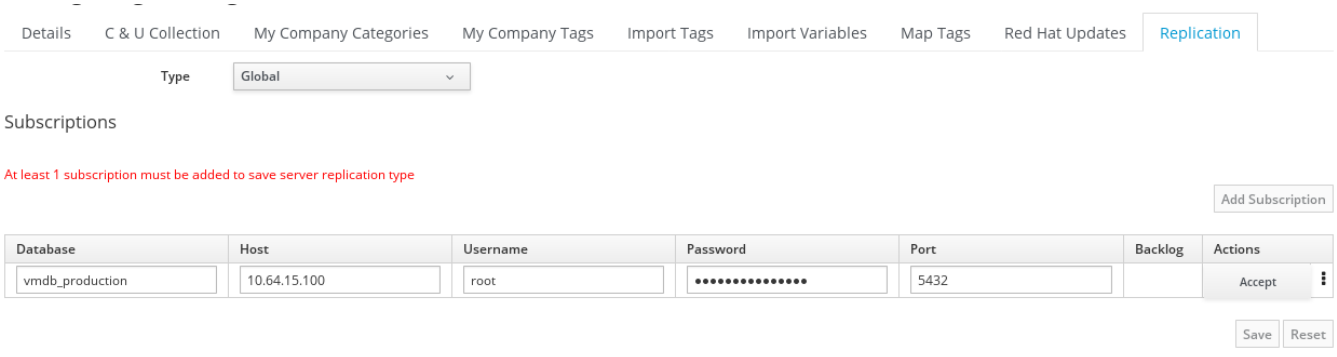
Configuring a Remote Copy

Configure a Infrastructure Automation instance to act as a remote copy from which data will be replicated to the global copy.

1. Browse to **Settings > Application Settings**.
2. Click the **Settings** accordion.
3. Click the region where the instance is located.
4. Click **Replication**.
5. Select **Remote** from the **Type** list.
6. Click **Save**.

Configuring the Global Copy

Configure a Infrastructure Automation instance to act as the global copy to which data is replicated from the remote copies.



1. Click **Configuration**.
2. Click the **Settings** accordion.
3. Click the region where the instance is located.
4. Click **Replication**.
5. Select **Global** from the **Type** list.

6. Click **Add Subscription**.
    1. Enter the name of the database on the remote copy in the **Database** field.
    2. Enter the IP address or fully qualified domain name of the remote copy in the **Host** field.
    3. Enter the name of a database user able to access the database in the **Username** field.
    4. Enter the password of the database user in the **Password** field.
    5. Enter the port by which the database is accessed in the **Port** field.
    6. In **Actions**, click **Accept**. You can also **Update** your subscription if required, or click the menu on the right in order to **Validate** or **Discard**.
7. Click **Save**.

Note:

Once you configure a Infrastructure Automation instance to act as a global copy, and one or more other instances to act as remote copies: * centralized administration is automatically enabled after the initial data sync is complete. * you will see the type of region you are logging into (global or remote) on the Infrastructure Automation user interface login screen.

Database replication and centralized administration are now enabled on your instances. To open virtual machines of a remote region from the global region and perform required operations:

1. Browse to menu: **Compute < Infrastructure < Virtual Machines**, then click **VMs & Templates** and select a virtual machine in the global region.
2. In the **Multi Region** table on the summary screen, click **Connect to VM in its Region** to perform required operations in a new browser tab.

Resetting Database Replication

You can reset the replication relationship between the global copy and remote copies by temporarily removing and re-enabling the subscription from the global copy.

1. Browse to **Settings > Application Settings**.
2. Click the **Settings** accordion.
3. Click the region where the instance is located.
4. Click **Replication**.
5. Remove the subscription:
    1. Click the actions button for the subscription to reset.
    2. Click **OK**.
6. Click **Save**.
7. Re-add the subscription:
    1. Click **Add Subscription**.
    2. Enter the name of the database on the remote copy in the **Database** field.
    3. Enter the IP address or fully qualified domain name of the remote copy in the **Host** field.
    4. Enter the name of a database user able to access the database in the **Username** field.
    5. Enter the password of the database user in the **Password** field.
    6. Enter the port by which the database is accessed in the **Port** field.
8. Click **Save**.

Backing Up and Restoring Infrastructure Automation in a Kubernetes Based Environment (podified)

Prerequisites:

- Velero and Restic must be installed and working in your cluster. For more information on Velero, see the Velero documentation. For more information on Restic, see the Restic documentation

Note: The operator adds labels and/or annotations to everything that needs to be backed up. You can easily and quickly backup only what is required. The operator will add annotations for restic volume backups for example, **`backup.velero.io/backup-volumes: <volume-label>`**.

1. The backup label key can be configured by adding the following to the CR. The default label is applied if no value is specified for **`backupLabelName`**.

   ```
   ...
   spec:
     ...
     backupLabelName: <your label name>
   ```

2. Create a backup by running the command:

   ```
   velero backup create <your backup name> --include-namespaces <your namespace> -l <your label name>=t
   ```

3. In order to restore a backup, first ensure the Infrastructure Automation CRD exists. Then, restore the backup by running the commands:

   ```
   oc new-project <your namespace>
   velero restore create --from-backup <your backup name>
   ```

Backing Up and Restoring a Database for an Appliance Based Environment

Running a Single Database Backup

Infrastructure Automation supports database backups to **Network File System (NFS)**, **Samba**, **Amazon Web Service(AWS) S3** and **OpenStack Swift** storage.

Note:

Database backups to **AWS S3** require valid AWS credentials to perform this operation.

To run a single database backup:

1. Browse to **Settings > Application Settings**.
2. Click the **Diagnostics** accordion and click the **Region** name.
3. Click the **Database** tab.
4. If you have created a backup schedule and want to use the same depot settings, select the schedule in the **Backup Schedules** box.
5. If you do not want to use the settings from a backup schedule, select a type of server for storing the backups from the Type drop-down list in the **Database Backup Settings** box. You can use **Network File System** (NFS), **Samba** or **AWS S3**.
   - If you select **Samba**, enter the **Depot Name**, **URI**, **Username**, and a valid **Password**. Click **Validate** to check the settings.
   - If you select **AWS S3**, enter the **Depot Name**, **URI**, and select the **AWS Region** from the drop-down list. Enter the **Username**, and a valid **Password**. Click **Validate** to check the settings.
   - If you select **OpenStack Swift**, enter the **Depot Name**, **URI**, **Security Protocol** and **API Port**. Enter the **Username**, and a valid **Password**. Click **Validate** to check the settings.
   - If you select **Network File System**, enter the **URI**.
6. Click **Submit** to run the database backup.

Restoring a Database from a Backup

If a database is corrupt or fails, restore it from a backup. You can restore a backup from a local file, NFS, or Samba.

To restore a database from a backup:

1. Save the database backup file as **`/tmp/evm_db.backup`**. Infrastructure Automation looks specifically for this file when restoring a database from a local backup.
2. If you are restoring a database backup on a high availability environment, stop the replication manager service (**`repmgr13.service`**). This is not required in other Infrastructure Automation configurations.

   ```
   # systemctl stop repmgr13.service
   ```

3. Log in to the appliance as the **root** user.
4. Enter **`appliance_console`**, and press **Enter**.
5. Select **`Stop EVM Server Processes`** to stop processes on all servers that connect to this VMDB.

6. Enter **Y** to confirm.
7. After all processes are stopped, press **Enter** to return to the menu.
8. Press **Enter** again to manually configure settings.
9. Select **Restore Database From Backup**, then specify the location to restore the backup from in the **Restore Database File** menu:
    1. If you saved the database backup file locally as **/tmp/evm_db.backup**, select **Local file**. You can also restore from a **Network File System (nfs)** or **Samba (smb)**.
    2. Specify the location of the backup file.

       Note:

       The appliance console menu may respond slowly if connections are open and the server is still shutting down. If this occurs, wait a minute and try again.
10. Enter **Y** to keep the database backup after restoring from it. Enter **N** to delete it.
11. Press **Y** to confirm.
12. After the backup completes, press **Enter** to return to the menu.
13. Press **Enter** again to manually configure settings.
14. Select **Start EVM Server Processes** to restart all processes on servers that connect to this VMDB.
15. Enter **Y** to confirm.
16. If you are restoring a database backup on a high availability environment, start the replication manager service (**repmgr13.service**). This is not required in other Infrastructure Automation configurations.

    ```
    # systemctl start repmgr13.service
    ```

Performing a Binary Backup and Restoring the Database

Preserve data at the file system level by performing a binary backup. This includes all databases, users and roles, and other objects.

Note:

This procedure uses the **pg_basebackup** utility to perform a remote database backup and create a full replacement of the PostgreSQL data directory, capturing the exact state of the database when the backup finishes. For more information on the **pg_basebackup** utility, see the PostgreSQL documentation.

Performing a Binary Backup

Create a binary backup and store it as a **gzip** tar file inside the Infrastructure Automation backup directory.

A PostgreSQL superuser or user with *Replication* permissions are required to perform this procedure.

1. SSH into the database server as the **root** user or provide PostgreSQL superuser credentials.
2. Run the **pg_basebackup** command to create the backup:

   ```
   # pg_basebackup -X fetch -h hostname -U root -Ft -z -D filename
   ```

   Note:

   -h *hostname* specifies the IP address of the database server.

   -D *filename* specifies the name of the directory created to contain the backup.

Restoring a Database from the Binary Backup

Restore your PostgreSQL binary backup using the following steps. This process will require stopping both EVM and PostgreSQL services before restoring data.

1. Copy the existing backup to the target VM:

   ```
   # scp filename/base.tar.gz root@hostname:/var/www/miq
   ```

2. SSH to the target VM:

   ```
   # ssh root@hostname
   ```

3. Stop both the EVM and PostgreSQL servers:

```
# systemctl stop evmserverd
# systemctl stop $APPLIANCE_PG_SERVICE
```

4. Rename the existing data directory:

```
# mv /var/lib/pgsql/data /var/lib/pgsql/data.backup
```

5. Create a clean data directory:

```
# mkdir /var/lib/pgsql/data
```

6. Unzip the tar file to the new directory:

```
# tar -xzf /var/www/miq/base.tar.gz -C /var/lib/pgsql/data
```

7. Correct permissions:

```
# chown postgres:postgres /var/lib/pgsql/data
# chmod 700 /var/lib/pgsql/data
```

8. Restart the PostgreSQL and EVM servers:

```
# systemctl start $APPLIANCE_PG_SERVICE
# systemctl start evmserverd
```

Running Database Garbage Collection

The database server collects garbage automatically, but Red Hat may occasionally direct you to run database garbage collection manually in order to reclaim unused space in your VMDB.

To run database garbage collection:

1. Browse to **Settings > Application Settings**.
2. Click the **Diagnostics** accordion and click the **Region** name.
3. Click the **Database** tab.
4. In the **Run Database Garbage Collection Now** box, click **Submit**.

Changing the Database Password

Use the following procedures to change the password on appliances containing an internal database, and on worker appliances.

To change the password for an external database, use the procedure in Changing the Password on the Worker Appliances.

Note:

See Appliance Types in the *Deployment Planning Guide* for a summary of different types of appliances.

Changing the Password on the Database Appliance

Infrastructure Automation provides a default database password for the internal PostgreSQL database.

To change the password, you need to stop the Infrastructure Automation service, change the password for the PostgreSQL database, run a command to change the password in the configuration file that **evmserverd** uses to access the server, and restart the Infrastructure Automation appliance.

1. Stop the Infrastructure Automation service:
    1. SSH into the appliance.
    2. To stop the Infrastructure Automation service, run the following command:

    ```
    service evmserverd stop
    ```

2. Use **pgadmin** to change the password for your Infrastructure Automation database (default is **vmdb_production**). If you do not have **pgadmin**, you can change the password by running:

```
psql -U root -d vmdb_production
```

    1. At the vmdb# prompt, type:

    ```
    ALTER USER root WITH PASSWORD 'newpassword';
    ```

    2. To exit **psql**, type:

```
\q
```

3. Change the password in the configuration file that **evmserverd** uses to access the server:

```
/var/www/miq/vmdb/tools/fix_auth.rb --databaseyml --password newpassword
```

4. Restart the Infrastructure Automation service:

```
service evmserverd start
```

5. Verify that you can log in to the Infrastructure Automation console.

Changing the Password on the Worker Appliances

1. Stop the Infrastructure Automation service:
    1. SSH into the appliance.
    2. To stop the Infrastructure Automation service, run the following command:

    ```
    service evmserverd stop
    ```

2. Change the password in the configuration file that **evmserverd** uses to access the server:

```
/var/www/miq/vmdb/tools/fix_auth.rb --databaseyml --password newpassword
```

3. Restart the Infrastructure Automation service:

```
service evmserverd start
```

In a high availability environment, if using the same PostgreSQL user for replication, you must also change the password in the `/var/lib/pgsql/.pgpass` file on every database node. Additionally, if the password for the user being used for region-to-region replication is changing, users must also change the password in the replication subscription screen. See [Configuring the Global Copy](#configuring-the-global-copy).

Adding a New Appliance to an Existing Region with a Non-default Password

1. Create the new appliance.
2. Start the appliance, but do not go into any of the configuration options, instead **SSH** into the new appliance.
3. In the **/var/www/miq/vmdb** directory, create a file called **REGION**. Its only contents should be the number of the Region that it is joining. (You could also just copy the **REGION** file from the VMDB appliance.)
4. Edit the **database.yml** file in the **/var/www/miq/vmdb** directory. (You may want to save from the original.)
    1. Replace the contents of the **“production”** section with the contents of the **“base”** section.
    2. Edit the **“host”** parameter to match the IP of the appliance hosting the VMDB.
    3. Save the new **database.yml**.
5. Run the following command to change the password in the configuration file that **evmserverd** uses to access the server:

```
/var/www/miq/vmdb/tools/fix_auth.rb --databaseyml --password newpassword
```

6. Restart the new worker appliance:

```
service evmserverd restart
```

Creating a Database Dump

Creating a database dump from the appliance console can be used to provide additional information when raising a support case. Options exist to generate a database dump excluding specified tables and to split the database dump into multiple parts.

Database dumps require name or location information based on the **Dump Output File Name** selection:

Local File	Location
Network File System (NFS)	Location
Samba (SMB)	Location
Amazon S3	Name
File Transfer Protocol (FTP)	Location
ftp to dropbox.redhat.com	Name

To create a database dump:

1. SSH into the appliance console as **root**.
2. Press any key to continue.
3. Type **appliance_console** and press **Enter**.
4. From the Advanced Settings, select option 5, **Create a Database Dump**.
5. Select a **Dump Output File Name** from the list.
6. Enter the location or name to save the dump file to.
7. (Optional) Select tables to exclude from the dump:
    1. Enter **Y** to exclude tables. Select **N** to skip the operation.
    2. Enter the table names in a space-separated list:

       ```
       metrics_* vim_performance_states event_streams
       ```
8. (Optional) Split the database dump output into multiple parts:
    1. Enter **Y** to split the database dump output. Select **N** to skip the operation.
    2. Enter the size to split by, ex **250M** or **1G**.

The database dump is created and sent to the specified location.

Enabling Events Syndication

You can enable events syndication in the **Advanced** settings:

1. From the menu, go to **Settings > Application Settings**.
2. Expand the **Settings** accordian. When you select the **Settings** accordion, you are automatically taken to the server list under **Zones**.
3. Select the zone where the EVM server is located, and select the EVM server.
4. Click the **Advanced** tab.

   You can click the **Advanced** tab at the zone level to set advanced configuration settings for the entire zone. Or, you can click on the server, and then click the **Advanced** tab to set advanced configuration settings at the server (appliance) level, which overrides zone-level settings.
5. Find the following configuration:

   ```
   :event_streams:
     :history:
       :keep_events: 6.months
       :purge_window_size: 1000
     :syndicate_events: false
   ```
6. Set the value for **:syndicate_events:** to **true**.
7. Click **Save**.

SmartProxies

The embedded SmartProxy can analyze virtual machines that are registered to a host and templates that are associated with a provider.

Installing the SmartProxy from the Console

The server comes with one SmartProxy version already available. It can also be installed on an ESX Server version 3.0.2, 3.5 or 4.

Contact Red Hat before installing a new SmartProxy on an ESX Server.
Requirements:

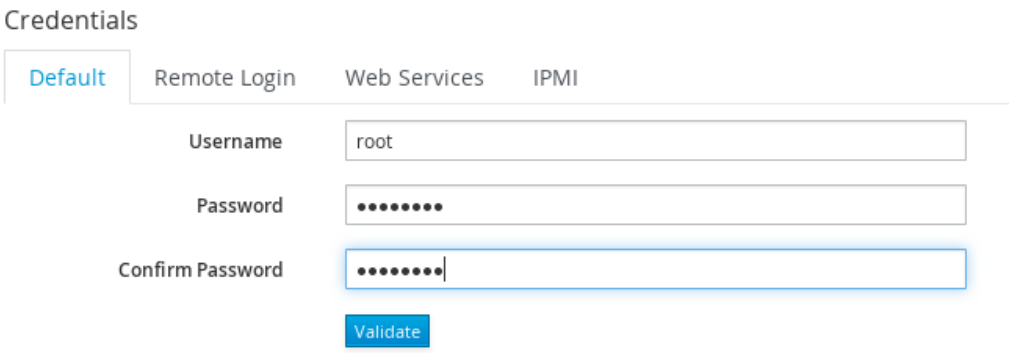
- On ESX, SSH (Secure Shell) must be enabled. This is usually port 22.
- 300 MB free disk space to install and run the SmartProxy.
- Administrator or root credentials.
- The host must already be in the VMDB either by discovery or manually.

Entering Credentials and Operating System for the Target Host

Set the credentials and operating system for the target host to prepare for the installation of SmartProxy.

To Enter Credentials and Operating System for the Target Host:

1. Browse to menu: **Compute > Infrastructure > Hosts**.
2. Select the host you want to edit.
3. Click **Configuration**, then **Edit Selected items**.
4. In **Credentials**, click the **Default** tab and enter your login credentials. If you are using domain credentials, the format for User ID must be in the format of `<domainname>\<username>`. For ESX hosts, if SSH login is disabled for the default user, click the **Remote Login** tab and enter a user with remote login access.



   If the target is a Windows host, disconnect all network connections between the Windows proxy and the target. If an existing connection uses a different set of credentials than those set in the console, the installation may fail.
5. Click **Validate** to verify the credentials.
6. If you added the host manually instead of **Host Discovery** or **Provider Refresh** finding it, select the host's operating system from the **Host Platform** drop-down box to ensure the host platform is available.
7. Click **Save**.

When remotely installing on Windows hosts, the SmartProxy file is first copied to a Windows proxy. That computer then installs the file to the target host. The Windows proxy is the same as when you select the Default Repository SmartProxy box. You can locate this by navigating to the settings menu and selecting **Configuration**, then clicking on the desired server, then the **Server** tab, and exploring the **Server Control** area.

About

Click the question mark icon ( ) at the top right of the Infrastructure Automation user interface to open a dropdown menu with links to Infrastructure Automation documentation and the Red Hat Customer Portal, and for information about the current Infrastructure Automation session.

- Click **Documentation** to open a list of guides. Click the title of a document to view it as a PDF in a new web browser tab.
- Click **Red Hat Customer Portal** to open the Red Hat Customer Portal in a new web browser tab.
- Click **About** to open a pop-up window showing information about the Infrastructure Automation version and the session of the logged-in user.

Default Roles

This section outlines the default roles provided in Infrastructure Automation and the product features to which they are granted access.

EvmRole-super_administrator

The `EvmRole-super_administrator` is granted access to the following product features.

Product Feature
everything

EvmRole-super_administrator

EvmRole-administrator

The **`EvmRole-administrator`** is granted access to the following product features.

Product Feature
about
all_vm_rules
automation_manager
embedded_automation_manager
availability_zone
host_aggregate
compute
flavor
floating_ip
chargeback
catalog
cloud_network
cloud_object_store_container
cloud_object_store_object
control_explorer
dashboard
datacenter
storage
storage_pod
ems_cloud
ems_network
ems_cluster
ems_infra
ems_infra_admin_ui
ems_physical_infra
container_dashboard
ems_container
ems_container_deployment
configuration_job
container_project
container_route
container_service
container_replicator
container_group
container
container_node
persistent_volume
container_build
container_image_registry
container_image
container_topology
cloud_topology
infra_topology
host
infra_networking
miq_ae_class_explorer
miq_ae_customization_explorer
miq_ae_class_import_export

Product Feature
miq_ae_class_log
miq_ae_class_simulation
miq_task_all_ui
miq_task_my_ui
my_settings
policy_simulation
policy_log
miq_report
miq_request
miq_template
orchestration_stack
physical_server
physical_infra_topology
planning
policy_import_export
provider_foreman_explorer
pxe
resource_pool
security_group
service
timeline
usage
utilization
vm_explorer
vm
vm_cloud_explorer
vm_infra_explorer
sui_services
sui_notifications

EvmRole-administrator

EvmRole-approver

The **`EvmRole-approver`** is granted access to the following product features.

Product Feature
about
all_vm_rules
compute
chargeback
chargeback_reports
customization_template_view
iso_datastore_view
control_explorer
dashboard
ems_cluster_show
ems_cluster_show_list
ems_cluster_perf
ems_cluster_tag
ems_cluster_timeline
ems_infra_show
ems_infra_show_list
ems_infra_tag
ems_infra_timeline
ems_physical_infra_console
ems_physical_infra_tag

Product Feature
ems_physical_infra_view
ems_physical_infra_tag
host_show
host_show_list
host_perf
host_tag
host_timeline
my_settings_default_views
my_settings_time_profiles
my_settings_visuals
miq_report_run
miq_report_saved_reports
miq_report_schedules
miq_report_view
miq_request_control
miq_request_view
miq_task_my_ui
miq_template_check_compliance
miq_template_perf
miq_template_policy_sim
miq_template_show
miq_template_show_list
miq_template_snapshot
miq_template_tag
miq_template_timeline
physical_infra_topology_view
physical_server_view
policy_log
policy_simulation
pxe_image_type_view
pxe_server_view
resource_pool_show
resource_pool_show_list
resource_pool_tag
service_view
storage_show
storage_show_list
storage_pod_show
storage_pod_show_list
storage_perf
storage_tag
timeline
usage
vm_check_compliance
vm_console
vm_webmks_console
vm_cloud_explorer
vm_explorer
vm_infra_explorer
vm_vnc_console
vm_vmrc_console
cockpit_console
vm_perf
vm_policy_sim
vm_show
vm_show_list
vm_snapshot

Product Feature
vm_tag
vm_timeline
vm_chargeback
sui_services_view
sui_vm_details_view
sui_vm_console
sui_vm_web_console
sui_vm_tags
sui_orders_view
sui_notifications

EvmRole-approver

EvmRole-auditor

The **`EvmRole-auditor`** is granted access to the following product features.

Product Feature
about
automation_manager
embedded_automation_manager
compute
chargeback
chargeback_reports
customization_template_view
iso_datastore_view
control_explorer
dashboard
ems_cluster_view
ems_cluster_tag
ems_infra_view
ems_infra_tag
ems_infra_check_compliance
ems_physical_infra_console
ems_physical_infra_tag
ems_physical_infra_view
host_show
host_show_list
host_perf
infra_networking_view
infra_networking_tag
instance_view
instance_check_compliance
instance_policy_sim
instance_tag
image_view
image_check_compliance
image_policy_sim
image_tag
iso_datastore_view
host_view
host_check_compliance
host_tag
miq_task_my_ui
my_settings_default_views
my_settings_time_profiles
my_settings_visuals

Product Feature
miq_report_run
miq_report_saved_reports
miq_report_schedules
miq_report_view
miq_template_view
miq_template_check_compliance
miq_template_policy_sim
miq_template_snapshot_view
miq_template_tag
physical_infra_topology_view
physical_server_view
planning
policy_log
policy_profile
policy_simulation
pxe_image_type_view
pxe_server_view
resource_pool_show
resource_pool_show_list
resource_pool_tag
service_view
storage_show
storage_show_list
storage_perf
storage_tag
storage_pod_show
storage_pod_show_list
timeline
usage
utilization
vm_view
vm_check_compliance
vm_compare
vm_console
vm_drift
vm_webmks_console
vm_cloud_explorer
vm_explorer
vm_infra_explorer
vm_vnc_console
vm_vmrc_console
cockpit_console
vm_perf
vm_show
vm_show_list
vm_snapshot_view
vm_tag
vm_timeline
vm_chargeback
sui_services_view
sui_vm_details_view
sui_vm_console
sui_vm_web_console
sui_vm_tags
sui_notifications

EvmRole-auditor

EvmRole-desktop

The **`EvmRole-desktop`** is granted access to the following product features.

Product Feature
about
all_vm_rules
automation_manager
compute
dashboard
ems_physical_infra
miq_request_admin
miq_request_view
miq_template_clone
miq_template_compare
miq_template_drift
miq_template_edit
miq_template_refresh
miq_template_miq_request_new
miq_template_perf
miq_template_publish
miq_template_show
miq_template_show_list
miq_template_timeline
my_settings_default_views
my_settings_time_profiles
my_settings_visuals
physical_server
physical_infra_topology
physical_server_view
provider_foreman_explorer
vm_clone
vm_compare
vm_console
vm_webmks_console
vm_cloud_explorer
vm_explorer
vm_infra_explorer
vm_vnc_console
vm_vmrc_console
cockpit_console
vm_drift
vm_edit
vm_refresh
vm_reset
vm_guest_restart
vm_miq_request_new
vm_perf
vm_publish
vm_show
vm_show_list
vm_guest_shutdown
vm_start
vm_stop
vm_suspend
vm_pause
vm_shelve

Product Feature
vm_shelve_offload
vm_timeline
vm_chargeback
sui_services_view
sui_vm_details_view
sui_vm_console
sui_vm_web_console
sui_vm_start
sui_vm_stop
sui_vm_suspend
sui_orders_view
sui_orders_operations
sui_notifications

EvmRole-desktop

EvmRole-operator

The **`EvmRole-operator`** is granted access to the following product features.

Product Feature
about
all_vm_rules
automation_manager
embedded_automation_manager
compute
chargeback
chargeback_reports
dashboard
datastore
ems_cluster_analyze
ems_cluster_compare
ems_cluster_drift
ems_cluster_show
ems_cluster_show_list
ems_cluster_perf
ems_cluster_tag
ems_cluster_timeline
ems_infra_new
ems_infra_delete
ems_infra_discover
ems_infra_edit
ems_infra_refresh
ems_infra_scale
ems_infra_show
ems_infra_show_list
ems_infra_tag
ems_infra_timeline
ems_physical_infra_new
ems_physical_infra_console
ems_physical_infra_delete
ems_physical_infra_discover
ems_physical_infra_edit
ems_physical_infra_refresh
ems_physical_infra_tag
ems_physical_infra_view
physical_server_view

Product Feature
physical_infra_topology_view
host_new
host_analyze
host_compare
host_discover
host_drift
host_edit
host_refresh
host_show
host_show_list
host_perf
host_tag
host_timeline
my_settings_default_views
my_settings_time_profiles
my_settings_visuals
miq_report_run
miq_report_saved_reports
miq_report_schedules
miq_report_view
miq_task_my_ui
miq_template_analyze
miq_template_check_compliance
miq_template_compare
miq_template_drift
miq_template_edit
miq_template_perf
miq_template_refresh
miq_template_show
miq_template_show_list
miq_template_snapshot
miq_template_sync
miq_template_tag
miq_template_timeline
pxe
resource_pool_show
resource_pool_show_list
resource_pool_tag
service_view
provider_foreman_explorer
storage_delete
storage_scan
storage_show
storage_show_list
storage_perf
storage_tag
timeline
usage
vm_analyze
vm_check_compliance
vm_collect_running_processes
vm_compare
vm_console
vm_webmks_console
vm_cloud_explorer
vm_explorer
vm_infra_explorer

Product Feature
vm_vnc_console
vm_vmrc_console
cockpit_console
vm_drift
vm_edit
vm_perf
vm_refresh
vm_reset
vm_guest_restart
vm_show
vm_show_list
vm_guest_shutdown
vm_snapshot
vm_start
vm_stop
vm_suspend
vm_pause
vm_shelve
vm_shelve_offload
vm_sync
vm_tag
vm_timeline
vm_chargeback
sui_services_view
sui_vm_details_view
sui_vm_console
sui_vm_web_console
sui_vm_tags
sui_vm_start
sui_vm_stop
sui_vm_suspend
sui_notifications

EvmRole-operator

EvmRole-security

The **`EvmRole-security`** is granted access to the following product features.

Product Feature
about
all_vm_rules
compute
chargeback
chargeback_reports
control_explorer
dashboard
datastore
ems_cluster_show
ems_cluster_show_list
ems_cluster_perf
ems_cluster_tag
ems_cluster_timeline
ems_infra_show
ems_infra_show_list
ems_infra_tag
ems_infra_timeline

Product Feature
ems_physical_infra_tag
ems_physical_infra_view
physical_server_timeline
host_show
host_show_list
host_perf
host_tag
host_timeline
my_settings_default_views
my_settings_time_profiles
my_settings_visuals
miq_report_run
miq_report_saved_reports
miq_report_schedules
miq_report_view
miq_task_my_ui
miq_template_check_compliance
miq_template_compare
miq_template_drift
miq_template_perf
miq_template_policy_sim
miq_template_show
miq_template_show_list
miq_template_snapshot_add
miq_template_snapshot_delete
miq_template_snapshot_delete_all
miq_template_snapshot_revert
miq_template_tag
miq_template_timeline
policy_log
policy_simulation
resource_pool_show
resource_pool_show_list
resource_pool_tag
service_view
storage_show
storage_show_list
storage_perf
storage_tag
timeline
usage
vm_check_compliance
vm_compare
vm_drift
vm_cloud_explorer
vm_explorer
vm_infra_explorer
vm_perf
vm_policy_sim
vm_show
vm_show_list
vm_snapshot_add
vm_snapshot_delete
vm_snapshot_delete_all
vm_snapshot_revert
vm_tag
vm_timeline

Product Feature
vm_chargeback
sui_services_view
sui_vm_details_view
sui_vm_snapshot_create
sui_vm_snapshot_delete
sui_vm_tags
sui_notifications

EvmRole-security

EvmRole-support

The **`EvmRole-support`** is granted access to the following product features.

Product Feature
about
all_vm_rules
compute
chargeback
chargeback_reports
control_explorer
dashboard
datastore
ems_cluster_show
ems_cluster_show_list
ems_cluster_perf
ems_cluster_tag
ems_cluster_timeline
ems_infra_show
ems_infra_show_list
ems_infra_tag
ems_infra_timeline
ems_physical_infra_console
ems_physical_infra_tag
ems_physical_infra_view
host_show
host_show_list
host_perf
host_tag
host_timeline
miq_task_my_ui
my_settings_default_views
my_settings_time_profiles
my_settings_visuals
miq_report_run
miq_report_saved_reports
miq_report_schedules
miq_report_view
miq_template_check_compliance
miq_template_compare
miq_template_drift
miq_template_perf
miq_template_policy_sim
miq_template_show
miq_template_show_list
miq_template_snapshot
miq_template_tag

Product Feature
miq_template_timeline
physical_infra_topology_view
physical_server_view
policy_log
policy_simulation
resource_pool_show
resource_pool_show_list
resource_pool_tag
service_view
storage_show
storage_show_list
storage_perf
storage_tag
timeline
usage
vm_check_compliance
vm_collect_running_processes
vm_console
vm_webmks_console
vm_cloud_explorer
vm_explorer
vm_infra_explorer
vm_vnc_console
vm_vmrc_console
cockpit_console
vm_compare
vm_drift
vm_perf
vm_policy_sim
vm_show
vm_show_list
vm_snapshot
vm_tag
vm_timeline
vm_chargeback
sui_services_view
sui_vm_details_view
sui_vm_console
sui_vm_web_console
sui_vm_tags
sui_notifications

EvmRole-support

EvmRole-user

The **`EvmRole-user`** is granted access to the following product features.

Product Feature
about
all_vm_rules
compute
chargeback
chargeback_reports
dashboard
datastore
ems_cluster_show

Product Feature
ems_cluster_show_list
ems_cluster_perf
ems_cluster_tag
ems_cluster_timeline
ems_infra_show
ems_infra_show_list
ems_infra_tag
ems_infra_timeline
ems_physical_infra_console
ems_physical_infra_show
ems_physical_infra_show_list
ems_physical_infra_tag
ems_physical_infra_timeline
host_show
host_show_list
host_perf
host_tag
host_timeline
miq_task_my_ui
my_settings_default_views
my_settings_time_profiles
my_settings_visuals
miq_report_run
miq_report_saved_reports
miq_report_schedules
miq_report_view
miq_request_admin
miq_request_view
miq_template_check_compliance
miq_template_compare
miq_template_drift
miq_template_perf
miq_template_show
miq_template_show_list
miq_template_snapshot
miq_template_tag
miq_template_timeline
physical_infra_topology_view
physical_server_view
resource_pool_show
resource_pool_show_list
resource_pool_tag
service_view
storage_show
storage_show_list
storage_perf
storage_tag
timeline
usage
vm_check_compliance
vm_console
vm_webmks_console
vm_cloud_explorer
vm_explorer
vm_infra_explorer
vm_vnc_console
vm_vmrc_console

Product Feature
cockpit_console
vm_compare
vm_drift
vm_perf
vm_show
vm_show_list
vm_snapshot
vm_tag
vm_timeline
vm_chargeback
sui_services_view
sui_vm_details_view
sui_vm_console
sui_vm_web_console
sui_vm_tags
sui_orders_view
sui_orders_operations
sui_notifications

EvmRole-user

EvmRole-user_limited_self_service

The **`EvmRole-user_limited_self_service`** is granted access to the following product features.

Product Feature
about
all_vm_rules
catalog_items_view
compute
miq_request_admin
miq_request_view
my_settings_default_views
my_settings_visuals
service_edit
service_delete
service_reconfigure
service_tag
service_retire_now
service_view
svc_catalog_provision
vm_clone
vm_cloud_explorer
vm_filter_accord
vm_guest_restart
vm_guest_shutdown
vm_infra_explorer
vm_miq_request_new
vm_publish
vm_reset
vm_retire_now
vm_show
vm_show_list
vm_snapshot_view
vm_start
vm_stop
vm_suspend

Product Feature
vm_pause
vm_shelve
vm_shelve_offload
vm_tag
sui_core
sui_services
sui_vm
sui_orders
sui_svc_catalog_view
sui_svc_catalog_cart
sui_cart

EvmRole-user_limited_self_service

EvmRole-user_self_service

The **`EvmRole-user_self_service`** is granted access to the following product features.

Product Feature
about
all_vm_rules
automation_manager
embedded_automation_manager
ems_physical_infra_console
catalog_items_view
compute
miq_template_clone
miq_template_drift
miq_template_edit
miq_template_perf
miq_template_refresh
miq_template_show
miq_template_show_list
miq_template_snapshot
miq_template_sync
miq_template_tag
miq_template_timeline
my_settings_default_views
my_settings_visuals
miq_request_admin
miq_request_view
provider_foreman_explorer
service_edit
service_delete
service_reconfigure
service_tag
service_retire_now
service_view
svc_catalog_provision
vm_console
vm_webmks_console
vm_clone
vm_cloud_explorer
vm_console
vm_webmks_console
vm_drift
vm_edit

Product Feature
vm_filter_accord
vm_guest_restart
vm_guest_shutdown
vm_infra_explorer
vm_miq_request_new
vm_perf
vm_publish
vm_refresh
vm_reset
vm_retire_now
vm_show
vm_show_list
vm_snapshot
vm_start
vm_stop
vm_suspend
vm_pause
vm_shelve
vm_shelve_offload
vm_sync
vm_tag
vm_timeline
vm_chargeback
vm_vmrc_console
vm_vnc_console
cockpit_console
sui

EvmRole-user_self_service

EvmRole-vm_user

The **`EvmRole-vm_user`** is granted access to the following product features.

Product Feature
about
all_vm_rules
automation_manager
embedded_automation_manager
compute
miq_request_admin
miq_request_view
miq_template_analyze
miq_template_check_compliance
miq_template_clone
miq_template_drift
miq_template_edit
miq_template_perf
miq_template_show
miq_template_show_list
miq_template_snapshot
miq_template_sync
miq_template_tag
miq_template_timeline
my_settings_default_views
my_settings_visuals
provider_foreman_explorer

Product Feature
vm_analyze
vm_check_compliance
vm_clone
vm_cloud_explorer
vm_collect_running_processes
vm_compare
vm_console
vm_webmks_console
vm_drift
vm_explorer
vm_guest_restart
vm_guest_shutdown
vm_infra_explorer
vm_miq_request_new
vm_perf
vm_policy_sim
vm_publish
vm_refresh
vm_reset
vm_retire_now
vm_show
vm_show_list
vm_snapshot
vm_start
vm_stop
vm_suspend
vm_pause
vm_shelve
vm_shelve_offload
vm_sync
vm_tag
vm_timeline
vm_chargeback
vm_vmrc_console
vm_vnc_console
cockpit_console
sui_vm_details_view
sui_vm_console
sui_vm_web_console
sui_vm_tags
sui_vm_retire
sui_vm_start
sui_vm_stop
sui_vm_suspend
sui_orders_view
sui_orders_operations
sui_notifications

EvmRole-vm_user

EvmRole-tenant_administrator

The **`EvmRole-tenant_administrator`** is granted access to the following product features.

Product Feature
about
automation_manager

Product Feature
embedded_automation_manager
availability_zone
host_aggregate
all_vm_rules
compute
cloud_network
cloud_subnet
flavor
floating_ip
chargeback
catalog
cloud_tenant
control_explorer
dashboard
datacenter
storage
ems_cloud
ems_network
ems_cluster
ems_infra
ems_physical_infra
host
load_balancer
miq_ae_class_explorer
miq_ae_customization_explorer
miq_ae_class_import_export
miq_ae_class_log
miq_ae_class_simulation
miq_task_all_ui
miq_task_my_ui
my_settings
network_port
network_router
policy_simulation
policy_log
miq_report
miq_request
miq_template
orchestration_stack
planning
policy_import_export
provider_foreman_explorer
pxe
rbac_group
rbac_role_view
rbac_tenant
rbac_user
resource_pool
security_group
service
timeline
usage
utilization
vm_explorer
vm
vm_cloud_explorer
vm_infra_explorer

Product Feature
sui_services
sui_notifications

EvmRole-tenant_administrator

EvmRole-tenant_quota_administrator

The **`EvmRole-tenant_quota_administrator`** is granted access to the following product features.

Product Feature
about
automation_manager
embedded_automation_manager
availability_zone
host_aggregate
all_vm_rules
flavor
compute
chargeback
catalog
cloud_tenant
control_explorer
dashboard
datacenter
storage
ems_cloud
ems_cluster
ems_infra
ems_physical_infra
host
miq_ae_class_explorer
miq_ae_customization_explorer
miq_ae_class_import_export
miq_ae_class_log
miq_ae_class_simulation
miq_task_all_ui
miq_task_my_ui
my_settings
policy_simulation
policy_log
miq_report
miq_request
miq_template
orchestration_stack
planning
policy_import_export
provider_foreman_explorer
pxe
rbac_tenant_view
rbac_tenant_manage_quotas
resource_pool
security_group
service
timeline
usage
utilization
vm_explorer

Product Feature
vm
vm_cloud_explorer
vm_infra_explorer
sui_services
sui_notifications

EvmRole-tenant_quota_administrator

EvmRole-consumption_administrator

The **`EvmRole-consumption_administrator`** is granted access to the following product features.

Product Feature
dashboard
chargeback
miq_report

EvmRole-consumption_administrator

EvmRole-container_administrator

The **`EvmRole-container_administrator`** is granted access to the following product features.

Product Feature
vms_filter_accord
instances_filter_accord
datacenter_controller
storage
storage_pod
dashboard
miq_report
consumption
chargeback
pictures
control_explorer
generic_object
generic_object_definition
my_settings
tasks
about
ems_container
container_group
container_node
container_replicator
container_image
container_image_registry
persistent_volume
container_build
container_template
container_service
container_route
container_project
container
container_topology
container_dashboard
ems_infra_dashboard

Product Feature
instance_view
vm_view
miq_cloud_networks
miq_arbitration_settings
miq_arbitration_rules
redhat_access_insights_admin
ems_container_ad_hoc_metrics
monitor
monitor_alerts
alert_status
alert_action
ems_infra
rbac_user
ops_settings

EvmRole-container_administrator

EvmRole-container_operator

The **`EvmRole-container_operator`** is granted access to the following product features.

Product Feature
vms_filter_accord
instances_filter_accord
dashboard_view
miq_report_saved_reports_view
miq_report_view
miq_report_control
chargeback_reports
my_settings
tasks
ems_container_view
ems_container_check_compliance
container_group_view
container_group_check_compliance
container_node_view
container_node_check_compliance
container_replicator_check_compliance
container_image_view
container_image_scan
container_image_check_compliance
container_image_registry_view
persistent_volume_view
container_build_view
container_template_view
container_service_view
container_route_view
container_project_view
container_filter_accord
container_view
container_control
container_topology
container_dashboard
instance_view
vm_view
ems_container_ad_hoc_metrics
monitor

Product Feature
monitor_alerts
alert_status
alert_action
ems_infra
rbac_user
ops_settings

EvmRole-container_operator

EvmRole-reader

The **`EvmRole-reader`** is granted access to the following product features.

Product Feature
auth_key_pair_cloud_view
automation_manager_configuration_script_view
automation_manager_configured_system_view
automation_manager_providers_view
availability_zone_view
catalog_items_view
cloud_network_view
cloudobject_store_container_view
cloud_object_store_object_view
cloud_subnet_view
cloud_tenant_view
cloud_topology
cloud_volume_backup_view
cloud_volume_snapshot_view
cloud_volume_view
configuration_job_view
configuration_script_view
configured_systems_filter_accord_view
container_build_view
container_dashboard
container_filter_accord
container_group_view
container_image_registry_view
container_image_view
container_node_view
container_project_view
container_replicator_view
container_route_view
container_service_view
container_template_view
container_topology
container_view
customization_template_view
dashboard_view
embedded_automation_manager_credentials_view
embedded_configuration_script_payload_view
embedded_configuration_script_source_view
ems_block_storage_view
ems_cloud_view
ems_cluster_view
ems_container_view
ems_infra_view
ems_physical_infra_view

Product Feature
ems_network_view
ems_object_storage_view
flavor_view
floating_ip_view
host_aggregate_view
host_view
image_view
infra_networking_view
infra_topology
instance_view
iso_datastore_view
load_balancer_view
miq_ae_class_log
miq_ae_domain_view
miq_report_saved_reports_view
miq_report_schedule_view
miq_report_view
miq_request_view
miq_template_snapshot_view
miq_template_view
my_settings
network_port_view
network_router_view
network_topology
orchestration_stack_view
orchestration_templates_view
persistent_volume_view
physical_infra_topology_view
physical_server_view
planning
policy_log
provider_foreman_view
pxe_image_type_view
pxe_server_view
rbac_group_view
rbac_role_view
rbac_tenant_view
rbac_user_view
redhat_access_insights_overview
resource_pool_view
security_group_view
service_view
st_catalog_view
storage_view
tasks
timeline
utilization
virtual_template_show
vm_cloud_explorer
vm_explorer
vm_infra_explorer
vm_snapshot_view
vm_view
sui_services_view
sui_orders_view
sui_notifications

EvmRole-reader

- Self Service User Interface Guide
- Infrastructure Automation Self Service
  - Accessing the Self Service User Interface
  - Navigating the Self Service User Interface
  - Dashboard Tab
  - My Services Tab
    - Setting a Retirement Date for a Service
    - Retiring a Service
    - Setting Ownership of a Service
    - Adding and Editing Tags
    - Removing a Service
    - Editing a Service
    - Controlling Virtual Machine Power States
    - Creating Virtual Machine Snapshots
    - Viewing Virtual Machine Snapshots
    - Accessing the Virtual Machine Console
    - Web Console (Cockpit) Integration
      - Accessing the Web Console
  - My Orders Tab
  - Service Catalog Tab

Self Service User Interface Guide

Infrastructure Automation Self Service

Infrastructure Automation Self Service is a web-based graphical user interface for ordering and managing IT service requests. You can enable self-service tenant end users, who can easily access their services, track requests, and manage their accounts using the Self Service user interface (SSUI), which has widgets, dashboard controls and feedback. The Self Service user interface supports role-based access control (RBAC) of menus and features, similar to in the full administrative user interface.

Note:

All catalog items, dialogs, buttons, icons among other items must be configured in the full administrative user interface before they are available to users in the Self Service user interface.

Accessing the Self Service User Interface

The Self Service user interface is running by default and does not need to be started or configured. To access the Self Service user interface, you must know the host name (or IP address) of the appliance, and the login credentials.

Self Service URL:

```
https://HOSTNAME/ui/service/
```

You can choose your preferred language for the Self Service user interface from the options in the **Language** list available on the login screen. The following languages are supported:

- English
- Español (Spanish)
- Français (French)
- 日本語 (Japanese)
- 한국어 (Korean)
- Nederlands (Dutch)
- Português (Portuguese)
- Português do Brasil (Brazilian Portuguese)
- 简体中文 (Simplified Chinese)
- Traditional Chinese

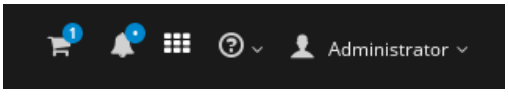
Additionally, the *User Default* and *Browser Default* options default to the language already set by the user or browser respectively.

Navigating the Self Service User Interface

The Self Service user interface has a slideout navigation menu on the left that can be minimized by clicking the menu icon (≡), like in the full administrative user interface. The slideout navigation menu comprises of the following tabs:

- **Dashboard**
- **My Services**
- **My Orders**
- **Service Catalogs**

The menu bar that runs along the top of the user interface contains several options including a shopping cart, notifications feature, full administrative user interface option, and a help option. Besides that, you can see the current logged in LDAP user, for example, **Administrator**.



- Shopping Cart Click the shopping cart icon to view services that are available to order, manage your cart as required, and order all services at one time. You can read more about the shopping cart model adopted in the Self Service user interface under ???.

- Notifications The Self Service user interface has introduced a new notifications feature that displays messages generated by the system for various events, such adding services to the shopping cart, ordering services, provisioning, service power operations, and so on. Click the bell icon to open or close the **Notifications** drawer, which shows various notification events. You can click the double arrows («) located on the upper-left of the **Notifications** drawer to expand or contract the size of the **Notifications** screen as required.
- Administration User Interface Link Clicking the menu icon ( ) shows the **Administration UI** link, which when clicked, takes you to the full administrative user interface login page in a new window or tab of your web browser.
- Help Click the help icon to access the options to open **Documentation**, **Red Hat Customer Portal**, and **About**. Click **About** to see information about your Self Service environment, such as the logged in user name and role, the browser details, and importantly, the Self Service user interface version number, which can be useful for support purposes.

Note:

The resources linked to in the help icon will be, by default, the ones listed above. System administrators can change them to other resources as required for the logged in user.

- Role-Based Access Control The Self Service user interface supports role-based access control of menus and features. If you are a member of multiple user groups, you can change the acting group by navigating to ***Administrator***

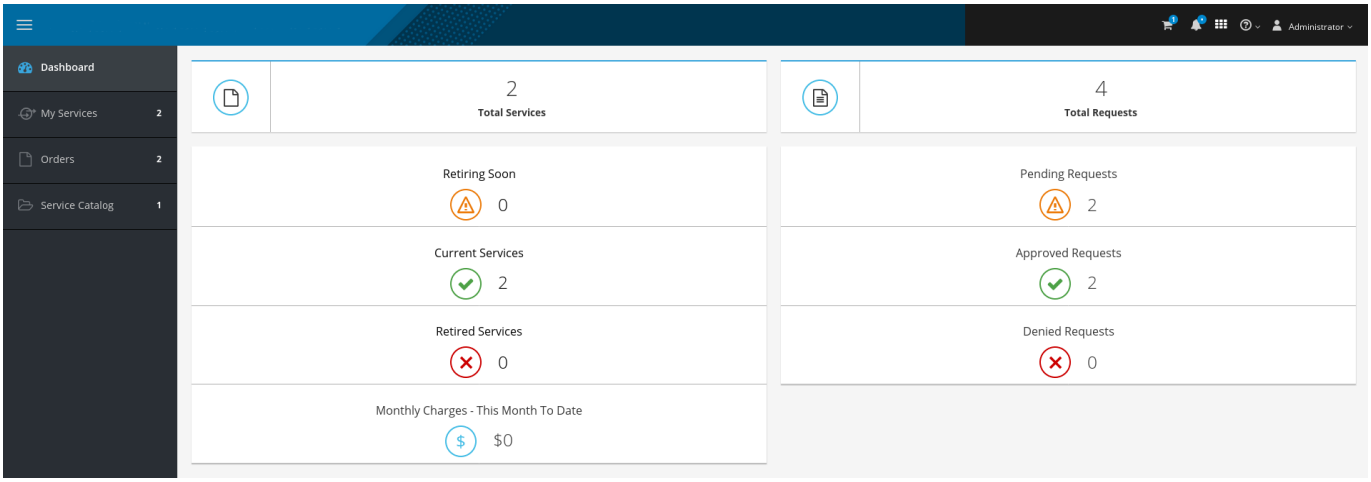
  Change Group, then select the new group from the available options in the list.
- Switch Language Although you can set your preferred language for the user interface from the login screen itself, you can switch to a different language than what is currently set also by navigating to ***Administrator***

  Switch Language within the Self Service user interface.

Dashboard Tab

Once you log in to the Self Service user interface, you will see the dashboard screen by default. The **Dashboard** tab provides a quick summary of the services available, requests made, and their statuses as below:

- Total Services
- Current Services
- Services Retiring Soon
- Retired Services
- Total Requests
- Pending Requests
- Approved Requests
- Denied Requests



- Monthly Charges - This Month To Date The Self Service user interface has added basic support for displaying monthly chargeback data of the computing and storage cost for the use of resources, in the currencies supported by the full administrative user interface. This provides a quick summary of where you are from cost standpoint to that point in the month. For more information about configuring chargeback rates in the full administrative user interface and how Infrastructure Automation calculates chargeback costs, see Chargeback in the *Monitoring, Alerts, and Reporting* guide.

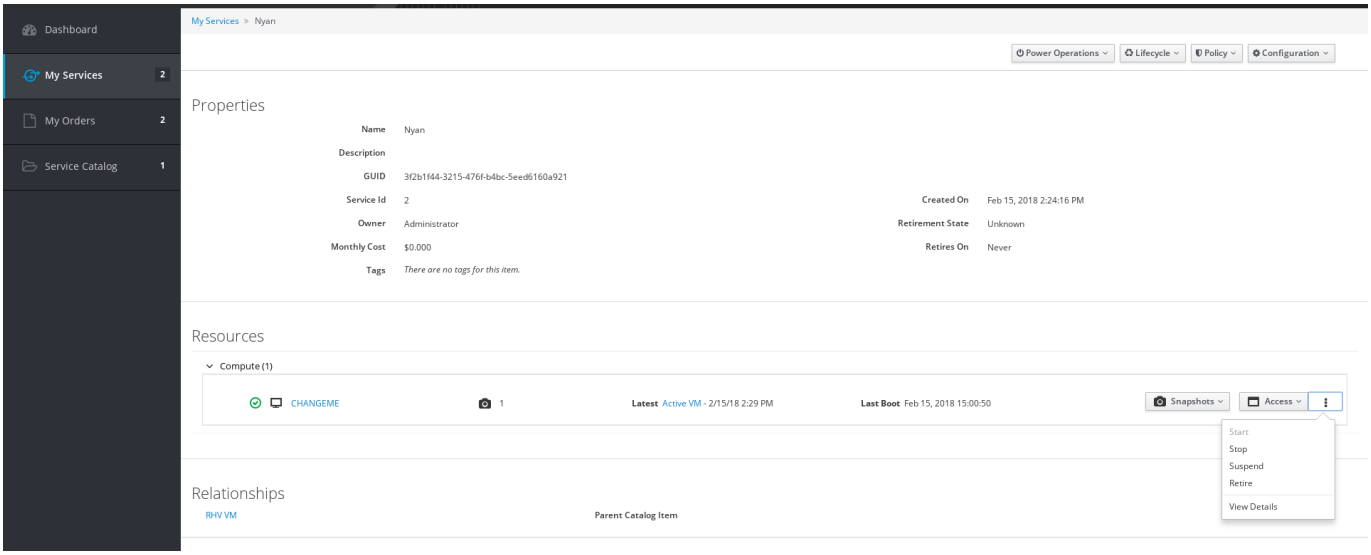
Note:

If you are on the first day of the month, there can be no data yet therefore resulting in zero dollar amount. Also, if you have a lot of data, it might not have rolled into the new month yet, resulting in no data and dollar amount for month to date.

My Services Tab

The **My Services** tab lists all service requests individually with a summary of information associated with each service, including the **Service Name**, its current **Power State** represented by an icon, **Number of VMs**, **Owner**, date **Created**, and **Retires** date. Additionally, you can see the chargeback cost per service for a virtual machine's month-to-date usage. You can filter or sort services by those entities, and by **Relative Cost**.

On the same screen, you have the option to edit the service name and description, edit tags, set ownership, retire the service now or at a later date, or remove the service by clicking more options *(represented by three dots)* next to the service. You also have options to perform power operations, that is start, stop, or suspend, at the service level. You can choose to click on a service where you can perform all of these actions at the service level using the designated buttons located at the top of the screen, as well as access the virtual machine dashboard, manage snapshots, and open the virtual machine and web console from under the **Resources** section. You can also control power operations and life cycle of virtual machine resources via more options *(represented by three dots)*.



Virtual Machine Dashboard:.

From the **My Services** screen, click on a service and then on the virtual machine name or **View Details** under more options *(represented by three dots)* to access the virtual machine dashboard. The dashboard displays the following information about the virtual machine resource:

- Compliance
- Retirement
- Snapshots
- Timeline of power events
- Utilization of CPU, Memory, and Storage
- Smart State analysis
- Tags

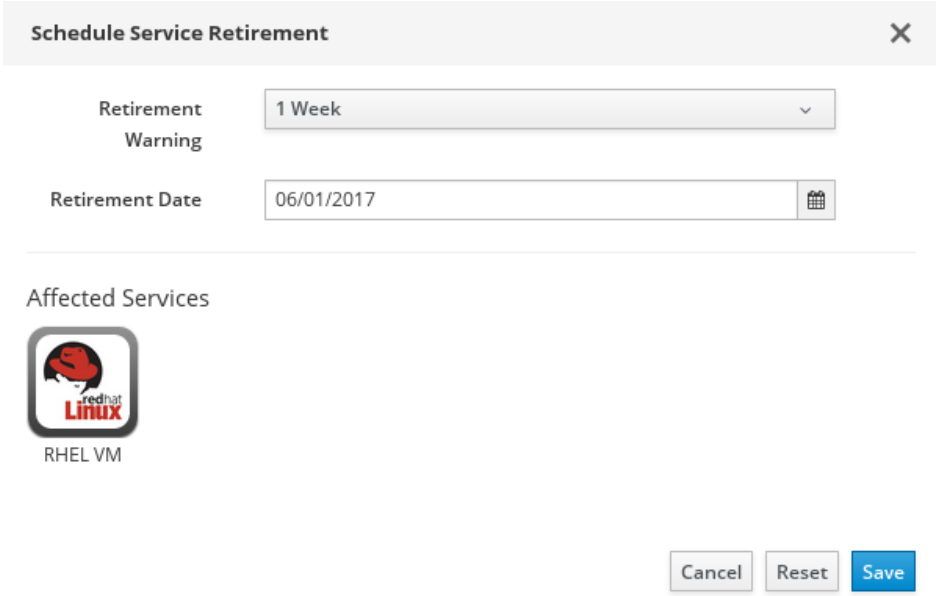
- Custom attributes
- Disk usage

Additionally, you can also control the virtual machine power operations, and manage snapshots from the dashboard. A snapshot is a copy of the virtual machine's disk file (VMDK) that captures the entire state of the virtual machine at a point in time.

Setting a Retirement Date for a Service

Follow this procedure to schedule retirement for a service at a later date. For retiring a service immediately, see Retiring a Service.

1. From the **Services** screen, click on the required service you want to schedule retirement for.
2. Click the button located on the upper-right corner of the screen, then select **Set Retirement Dates**. The **Schedule Service Retirement** window will appear.

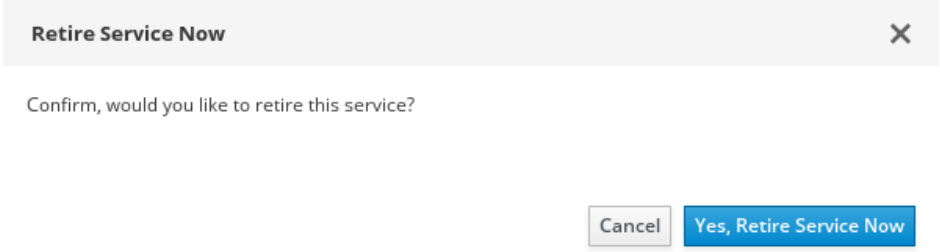


3. Set the **Retirement Date**.
4. Select the **Retirement Warning** from the list.
5. Click **Save**.

Retiring a Service

Follow this procedure to retire a service immediately. For retiring a service at a later date, see Setting a Retirement Date for a Service.

1. From the **Services** screen, click on the service you want to retire now.
2. Click the button located on the upper right corner of the screen, then select **Retire**. The **Retire Service Now** window will appear to confirm your action.



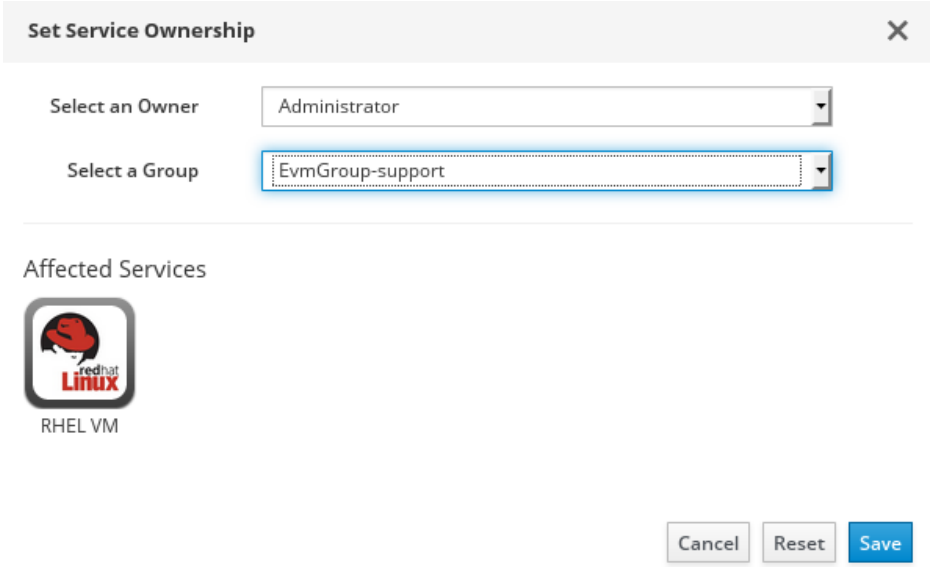
3. Click **Yes, Retire Service Now**.

Setting Ownership of a Service

Follow this procedure to set ownership of a service.

1. From the **Services** screen, click on the more options button next to the service you want set ownership for, then click **Set Ownership**.

   Alternatively, click on the service. Click the button located on the upper-right corner of the screen, then select **Set Ownership**.

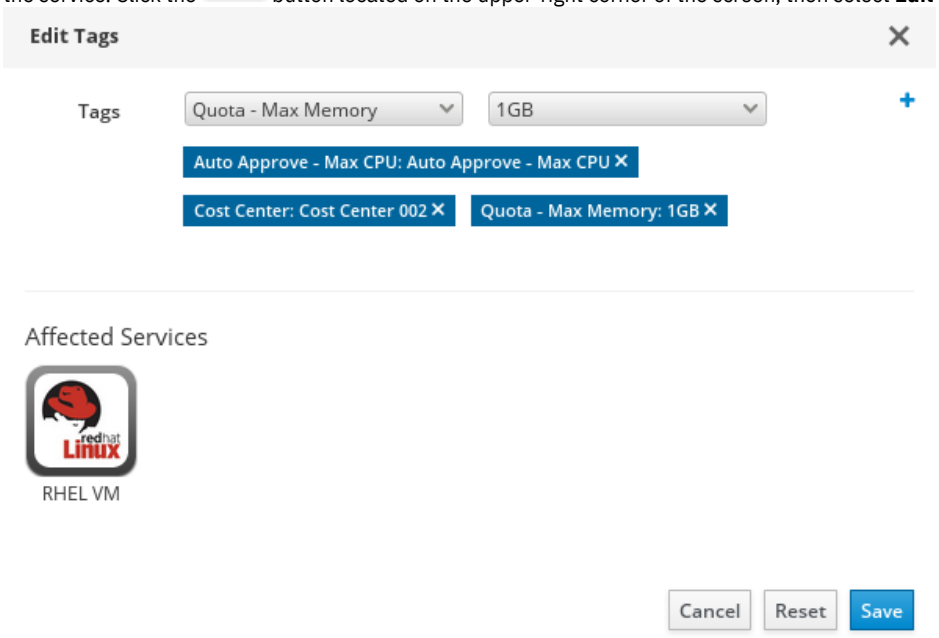


2. Select an owner from the list.
3. Select a group from the list.
4. Click **Save**.

Adding and Editing Tags

Follow this procedure to add or edit tags for a service.

1. From the **Services** screen, click on more options next to the service you want add or edit tags for, then click **Edit Tags**. Alternatively, click on the service. Click the button located on the upper-right corner of the screen, then select **Edit Tags**.



2. To add a tag, select the tag category and value from the lists, then click the blue plus (+) sign as shown in the screen capture.
3. To edit tags, simply change the value assigned to a category, or click on the cross sign in the tag to delete an existing tag.
4. Click **Save**.

Removing a Service

Follow this procedure to remove a service.

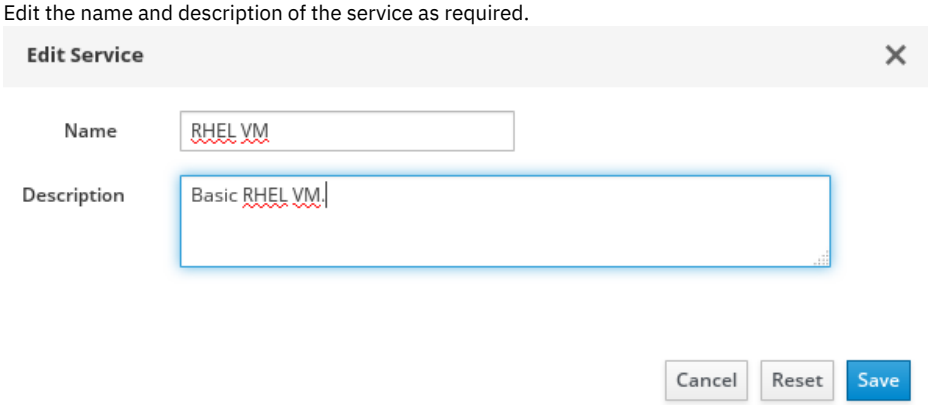
1. From the **Services** screen, click on more options next to the service you want to remove, then click **Remove**. Alternatively, click on the service. Click the button located on the upper-right corner of the screen, then select **Remove**. A confirmation window appears.

2. Click **Yes, Remove Service**.

Editing a Service

Follow this procedure to edit the name and description of a service.

1. From the **My Services** screen, click on more options next to the service you want to edit, then click **Edit**. Alternatively, click on the service. Click the button located on the upper-right corner of the screen, then select **Edit**. An **Edit Service** screen appears.
2. Edit the name and description of the service as required.



3. Click **Save**.

Controlling Virtual Machine Power States

Follow this procedure to perform power operations on virtual machines, such as start, stop, or suspend, depending on the current state of the virtual machine.

1. From the **My Services** screen, click on the required service.
2. Under **Resources**, click more options *(represented by three dots)* where you have the option to start, stop, or suspend the virtual machine. Alternatively, click on the virtual machine name to see more details about the resource, including the **Power Operations** button.

Creating Virtual Machine Snapshots

To create a snapshot from the Services screen:

1. From the **My Services** screen, click on the service associated with the virtual machine you want to create a snapshot for.
2. In **Resources**, click **Snapshots** next to the virtual machine you want to create a snapshot for, then select **Create** to create a snapshot. The **Create Snapshot** window appears.
3. Enter a name for the snapshot.
4. Optional: Select **Memory** if you want to capture the state of the virtual machine's memory.
5. Optional: Enter a **Description**.
6. Click **Create**.

To create a snapshot from the Resources screen:

1. From the **My Services** screen, click on the service associated with the virtual machine you want to create a snapshot for.
2. In **Resources**, click on the virtual machine you want to create a snapshot for.
3. On the virtual machine dashboard, click the **Snapshots** button, then select **Create**. The **Create Snapshot** window appears.
4. Enter a name for the snapshot.
5. Optional: Select **Memory** if you want to capture the state of the virtual machine's memory.
6. Optional: Enter a **Description**.
7. Click **Create**.

Viewing Virtual Machine Snapshots

1. From the **My Services** screen, click on the service associated with the virtual machine you want to view snapshots for.

2. In **Resources**, click **Snapshots** next to the virtual machine and select **View** to view available snapshots. Alternatively, click on the virtual machine, then click **Snapshots** to view available snapshots.

Note:

You can revert or delete a snapshot only from the snapshot summary screen. To do so, navigate to **Snapshots > View**, click more options *(represented by three dots)* next to the snapshot you want to revert or delete, and then click **Revert** or **Delete** as required. Revert returns a virtual machine to a previous state when you took the snapshot.

Accessing the Virtual Machine Console

Follow this procedure to open a console window to the virtual machine. The console is the remote control system that enables you to interact directly with your virtual machines.

1. From the **Services** screen, click on the required service.
2. In **Resources**, click the Access button next to the virtual machine and select **VM Console**.

The virtual machine console will open in a new window of your web browser. Note that you may have to enable the pop-up settings in your browser to access the console.

Web Console (Cockpit) Integration

The Web console (Cockpit) is an interactive Linux server administrative interface. You can perform simple administration tasks such as starting containers, storage administration, network configuration, and inspecting logs. While Cockpit allows you to monitor and administer several servers at the same time, your browser connects to a primary server that runs the Cockpit web service through which connections to additional servers can be established.

The Infrastructure Automation full administrative user interface and the Self Service user interface have enabled for users to access the Web console (Cockpit interface) for a given virtual machine.

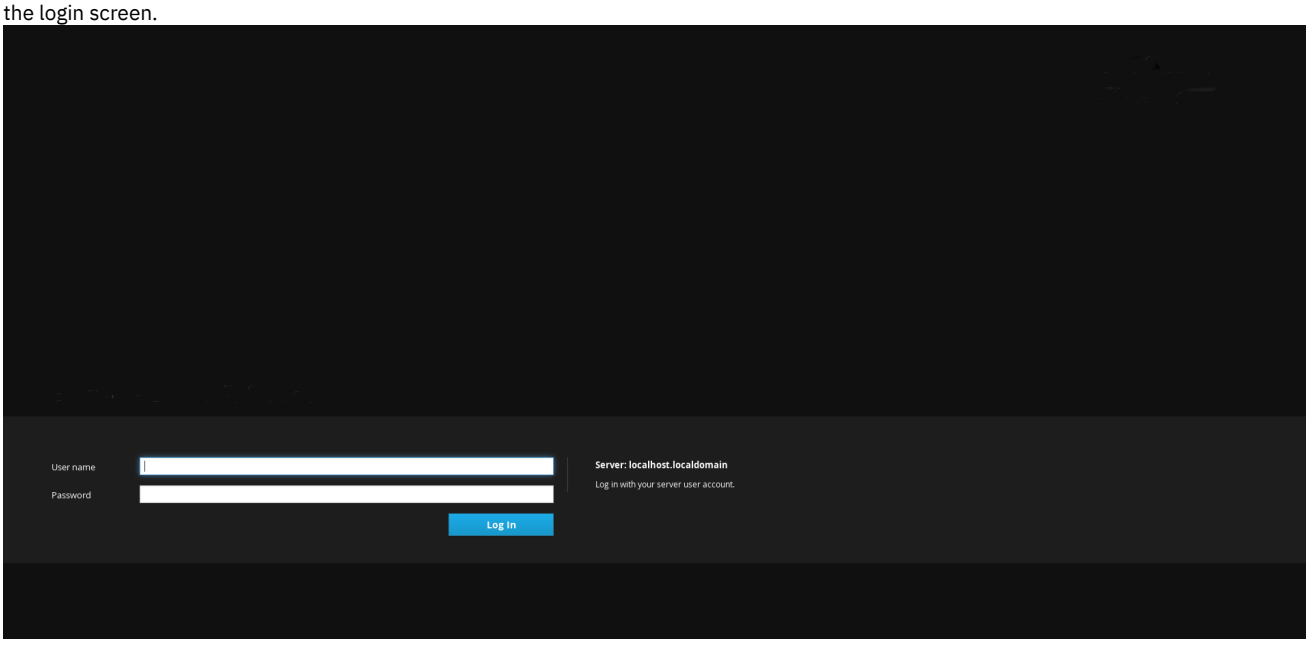
Accessing the Web Console

Note:

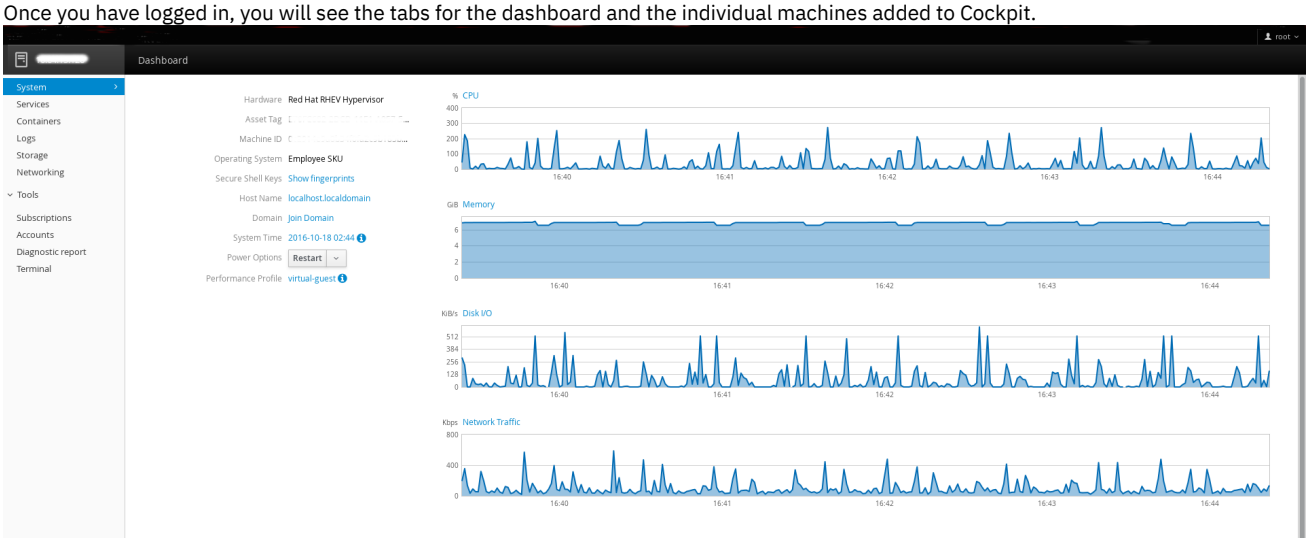
Launching of Cockpit requires as IP address for the virtual machine. It must be pre-configured and running on the virtual machine.

You can open the Web console (Cockpit) for a virtual machine from the Access button.

1. From the **My Services** screen, click on the service associated with the virtual machine you want to open the Web console for.
2. In **Resources**, click **Access** next to the virtual machine and select **Web Console**. A notification will appear to show the Cockpit interface is loading.
3. The Cockpit interface for the virtual machine will open in a new browser window. If you get a security warning by the browser, you will need to add this connection to the security exceptions. Click **Advanced → Add Exception → Confirm Security Exception**. After that, you will see the login screen.



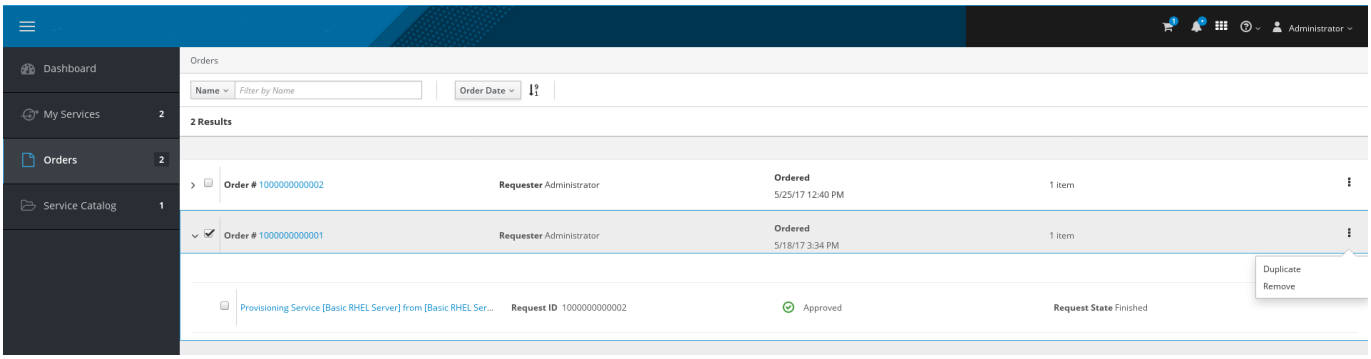
4. Once you have logged in, you will see the tabs for the dashboard and the individual machines added to Cockpit.



My Orders Tab

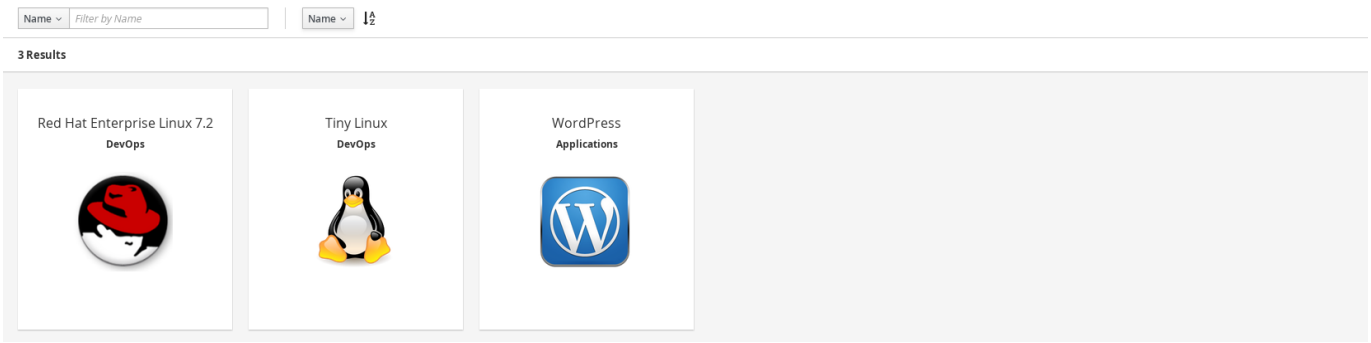
The **My Orders** tab lists all shopping cart orders which contain service requests added from the **Service Catalog**.

Click on the tab to see a summary of each order, including the order number, requester, order date, and the number of items. Click the arrow next to an order number to see more details such as the requests associated with the order, request ID, whether it is approved or denied, and the current state of the request. You can reorder a request by clicking the **Reorder** button, then enter the options associated with the order, and add to shopping cart. You can click on more options *(represented by three dots)* next to an order to **Remove** the order.



Service Catalog Tab

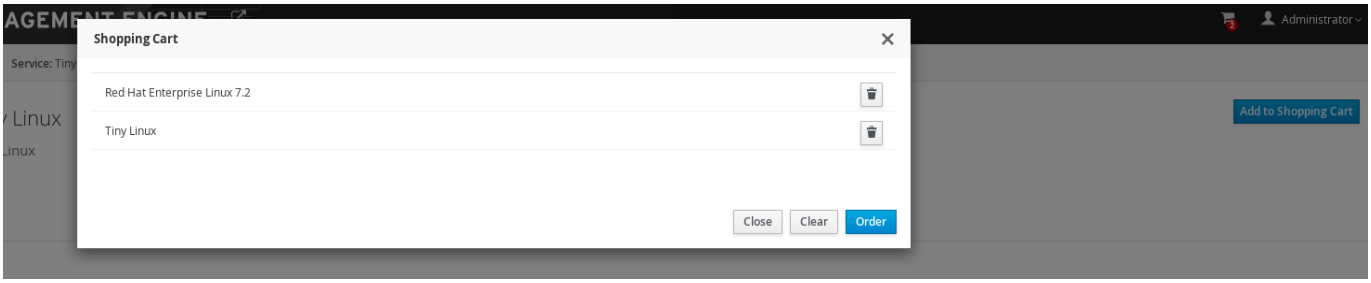
The **Service Catalog** tab lists all service catalog items, displayed as tiles. These items must be configured on the administrative user interface before they will be accessible to the user here. You can filter or sort the service catalog items by **Name**, **Description** and **Catalog Name**.



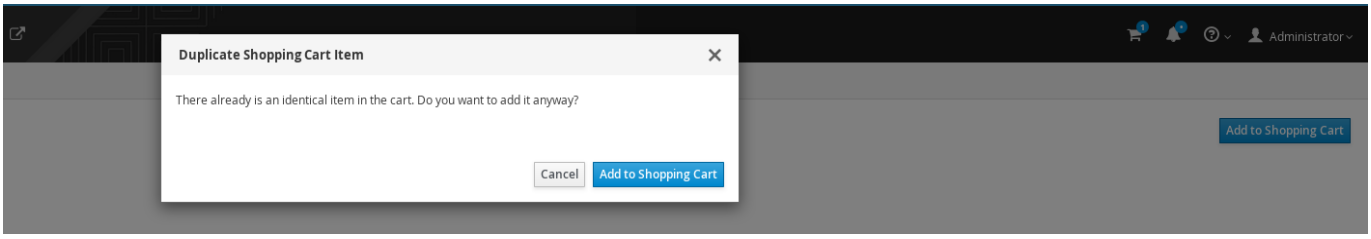
Click on a service to go to the page where you can enter the requested information in the dialog fields under **Service and VM Name** and **VM Characteristics** associated with the service. Note that there is now an ability to add dynamic dialog elements within the Self Service user interface, meaning you can include a particular dialog field where Infrastructure Automation will communicate to the instance and method, obtain the required information, and enter the value in the field dynamically. After you have entered all information in the dialog fields, click **Add to Shopping Cart**.

Shopping Cart:

The Self Service user interface has adopted a shopping cart model for ordering services, instead of ordering them individually. Click on the shopping cart icon at the top to view all services added to the shopping cart that are available to order. You can delete a particular service from the cart if required, and order the services in the cart all at one time. Or, you can even choose to clear all the services in the cart and exit.



If you click **Add to Shopping Cart** for the same item, you will see a pop up window alerting you that there is an identical item in the cart but if you want to add it anyway.



Reference

Go through the reference topics when you work with Managed services.

- Password migration of Managed services
- Managing the usage of virtual machines
- Managed services pods
- SSH Keys and related input required while deploying content runtime
- Control the version of a Terraform provider
- Terraform CAMC provider
- Shipped Terraform providers
- Supported Terraform templates

Password migration of Managed services

You can change the database passwords and encryption secrets of Managed services by using a password migration script.

Before you begin

- If you are planning to change the mariadb password, back up the mariadb as well.
- Download **`passwordMigration.sh`** from GitHub repository.

About the task

If you want to move to a different encryption password or change your bundled MongoDB password or mariadb password, use the **`passwordMigration`** tool. To change the encryption password, you reencrypt encrypted data in the database with the new encryption password.

Procedure

Run the following script:

```
./passwordMigration.sh [ -e ] [ -d ] [ -m ] [ -s ] [ -n ] [ -r ] [ -c ]
```

Flags to run the password migration: Usage:

- `-e` = new encryption password for Managed services. The flag changes the encryption password. Back up your MongoDB before using this).
- `-d` = new MongoDB password for Managed services. The flag changes the bundled MongoDB password. Back up your MongoDB before using this flag. It applies only for bundled MongoDB.
- `-m` = new mariadb password. This flag changes the bundled mariadb password. Back up your mariadb before using this flag. It applies only for bundled mariadb.
- `-s` = name of Managed services secret. The default is `cam-secure-values-secret`.
- `-n` = namespace for Managed services. The default is services.
- `-r` = Helm release name for Cloud Automation Manager
- `-c` = migration loop count to override the default timeout. The count is 300 by default.

Other flags can be left as default unless you changed values, such as cam secret name or namespace Managed services is installed into.

Here is an example usage to change encryption password, bundled MongoDB password, and bundled mariadb password:

```
./changePassword.sh -e newencpass -d newmongopass -m newmariapass 2>&1 | tee changePassword.log
```

When they run the password migration script, it stops Managed services. If Managed services is already running, it restarts Managed services after the script is successful. If password migration fails, you must restore your database(s).

Managing the usage of virtual machines in Managed services

You can keep a track of all virtual machines that are deployed on a system to arrive at the charges. The **Usage** page shows the highest number of virtual machines managed by Managed services each month. An average number of virtual Process Cores (VPC) per virtual machine can be entered to show the maximum and average VPCs managed each month.

About the task

You can see this option only if you log in as a cluster administrator.

The page fetches usage data only for the last 13 months.

To retrieve the currently deployed virtual machines, maximum number of virtual machines for each month, or to download the managed virtual machines report by using APIs, see Usage specific APIs.

Procedure

1. Click **Manage>Usage**.
2. View the **Current usage** section to get the number of currently actively deployed virtual machines.
3. In the **Usage** page, select your average VPCs per virtual machine. Based on the value entered, the VPCs managed is calculated. The formula used for the calculation is VMs managed * Average VPCs per VM. The monthly usage report table is populated based on your selection.
4. Go through the monthly usage table to get information about the month of usage, peak usage (maximum VMs and VPCs managed) for that month, and average usage (Average VMs managed and Average VPCs managed) for that month.
5. You can use the pagination drop-down to select the number of records you want to display in a page at a time. Use the **>** or **<** to go through the next or previous set of records.
6. You can click the **Download** link option on table to download the report in a spreadsheet format. The downloaded report includes the following information - ID, iaas resource id, name, id from provider, type from provider, provider, tenant id, namespace, action, and timestamp.

Managed services pods

List of Managed services pods and their functions.

- `cam-bpd-cds` - Cloud discovery service for dynamic parameter support in the service composer.
- `cam-iaas` - IaaS API server for cloud connections, template management, deploying instances, and more.
- `cam-mongo` - MongoDB for storing all Managed services data.
- `cam-orchestration` - Orchestration of services.
- `cam-portal-ui` - User interface for catalog of services.
- `cam-provider-helm` - Provider for deploying Helm charts from a service.
- `cam-provider-terraform` - Provider for deploying Terraform templates.
- `cam-proxy` - Proxies all requests from port 30000 to different microservices in Managed services.
- `cam-service-composer-api` - API server for composing services.
- `cam-service-composer-ui` - User interface for composing services.
- `cam-tenant-api` - API for tenant.

- `cam-ui-basic` - User interface for header (top bar), navigation, Templates, Connections, Email Configurations, Instances, Content runtimes, and getting started page.
- `cam-provider-bpm` - Provider for deploying Business Process Manager processes from a service.
- `cam-provider-ansible` - Provider for managing and deploying Ansible artifacts like credential, inventory, and job templates.
- `cam-controller-manager` - Managed services Kubernetes resource controller.

SSH Keys and related input required while deploying content runtime

Content runtime deployment requires you to input public and private keys. This section provides the purpose of these keys and how they are used by the content runtime.

User's Public Key

This is an SSH public key of the private-public key pair that you can generate using ssh-keygen tool. If you enter a value for User's Public Key while you deploy the content runtime, then it is set in the **`~/.ssh/authorized_keys`** file of the content runtime virtual machine. After the virtual machine is up, you can use the private key part of the public key to log in to the content runtime virtual machine.

For **`ssh-keygen`** tool, see the "Input Parameter" section of Provisioning and managing your Content Runtime infrastructure topic.

Private Key for Pattern Manager - Base 64 encoded, Public Key for Pattern Manager and Key Name for Pattern Manager key set

This is an SSH private-public key pair that you can generate by using the ssh-keygen tool. You must encode the private part of the key pair as a base 64 encoded value and provide this encoded value as input for **`Private Key for Pattern Manager - Base 64 encoded`**. The public part of this key pair must be the value for **`Public Key for Pattern Manager`**. The private key is used to communicate with the virtual machine on which middleware is installed. The public part of this key is set on this virtual machine while installing the middleware using the middleware content Terraform template.

In case of IBM Cloud and Amazon Cloud, the public key that needs to be set on the content virtual machine must be provided as SSH Key resource. For more information on SSH Key resource, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html and https://cloud.ibm.com/docs/ssh-keys?topic=ssh-keys-adding-an-ssh-key. This SSH Key resource must exist in the cloud prior to the deployment of the content runtime. The name of the SSH Key resource must be the value of content runtime input **`Key Name for Pattern Manager key set`**. When the content Terraform template is deployed, the key name is used to fetch the public key and the public key is set in the middleware content virtual machine.

Note: In case of VMWare and other clouds, the **`Key Name for Pattern Manager key set`** is just a name associated with the key pair.

Controlling the version of Terraform provider

In order to make use of certain functions and features of a Terraform provider, you might want to use a particular version of a Terraform provider. To use a particular version of a provider, add version argument in the provider block of Terraform. This prevents Terraform engine from using the latest version of the provider and instead use the version that you need.

The value of the version can be a single version or a range of versions. You can use the following formats to specify a range of versions:

- `>= 1.0.0` - Versions greater than or equal to the 1.0.0.
- `<= 1.0.0` - Versions lesser than or equal to 1.0.0.
- `>= 1.0.0, <= 2.0.0` - This includes all versions between 1.0.0 and version number 2.0.0
- `~> 1.0.0` - Any non-beta version greater than or equal to 1.0.0 and lesser than or equal to 1.1.0.
- `~> 1.1` - Any non-beta version greater than equal to 1.1.0 and lesser than 2.0

Example:

```
provider "aws" {
  version = "~> 1.0"
  access_key = "${var.aws_access_key}"
  secret_key = "${var.aws_secret_key}"
  region     = "${var.aws_region}"
}
```

Terraform CAMC provider

CAMC bootstrap

Description

The camc_bootstrap function performs a Chef bootstrap operation on the target node defined by the host_ip parameter. This function relies on a viable SSH connection between the pattern manager and the target node using SSH certificate-based security. The following steps are taken:

1. Install chef-client on the target node.
2. Register the Chef client with the Chef server. The Chef server will be bound to a single pattern manager as a parameter. If a Chef client already exists with the specified node_name parameter, then the target node is assumed to be a part of a previous deployment and is removed prior to registration.
3. Install the chef-vault gem.
4. Create a chef-vault entry based on the stack_id parameter of the node if it does not already exist. Chef nodes in a multi-node deploy will have the same stack_id and will share vault entries.

Example Usage

```
resource "camc_bootstrap" "resource1" {
    name = "resource1"
    camc_endpoint = "http://${var.endpoint}:5443/v1/bootstrap/"
    access_token = "${var.pm_access_token}"
    trace = "true"
    data = <<EOT
    {
        "os_username": "${var.referenced-image-os_username}",
        "stack_id": "${random_id.stack_id.hex}",
        "environment_name": "_default",
        "host_ip": "${vsphere_virtual_machine.referenced-image.network_interface.0.ipv4_address}",
        "node_name": "${var.referenced-image-name}",
        "node_attributes": {
            "ibm_internal": {
                "stack_id": "${random_id.stack_id.hex}",
                "stack_name": "${var.stack_name}",
                "vault": {
                    "item": "secrets",
                    "name": "${random_id.stack_id.hex}"
                }
            }
        }
    }
EOT
}
```

Argument Reference

Table 1. Parameters used with the camc_bootstrap provider

Type	Name	Description	Schema	Default	Required
Body	access_token	The pattern manager access token that was provided during Content Runtime provisioning	String		Yes
Body	camc_endpoint	The IP address or FQDN of the pattern manager	String		Yes
Body	data	A valid JSON structure. See *Data type* for more information	JSON		Yes
Body	name	A label to aid in identifying the resource	String		No
Body	trace	If true, writes more data to TF_LOG	Boolean	False	No
Data	os_username	The operating system user name used by the pattern manager to remote SSH to the target node	String		Yes
Data	stack_id	The stack ID shared across nodes in this deployment	String		Yes
Data	environment_name	The environment name for this deployment	String		Yes
Data	host_ip	The IP address of the host being bootstrapped	String		Yes
Data	node_name	The unique name to give the node on your chef server	String		Yes
Data	node_attributes	Chef attributes to be added to the node object during the bootstrap process.	String		Yes

CAMC software deploy

Description

Deploys one or more Chef cookbooks and Chef roles on the target node specified by the node_name parameter.

This node_name should match the value specified in the camc_bootstrap resource. The attributes specified by the node_attributes parameter will be applied to the node prior to executing the software deployment. Additionally, the vault may also be updated with the attributes defined in the vault_content parameter. The runlist parameter specifies an ordered list of roles and/or recipes to be run against the node. These cookbooks and roles need to be registered with the Chef server prior to executing this resource.

Example Usage

```
resource "camc_softwaredeploy" "resource2" {
    name = "resource2"
    camc_endpoint = "http://${var.endpoint}:5443/v1/softwaredeploy/chef"
    access_token = "${var.pm_private_b64}"
    data = <<EOT
    {
        "os_username": "${var.referenced-image-os_username}",
        "stack_id": "${random_id.stack_id.hex}",
        "environment_name": "_default",
        "host_ip": "${vsphere_virtual_machine.referenced-image.network_interface.0.ipv4_address}",
        "node_name": "${var.referenced-image-name}",
        "runlist": "role[ihs-wasmode-admin]",
        "node_attributes": {
            "host_ip": "${vsphere_virtual_machine.referenced-image.network_interface.0.ipv4_address}",
            "ibm": {
                "im_repo": "${var.ibm_im_repo}",
                "im_repo_user": "${var.ibm_im_repo_user}",
                "sw_repo": "${var.ibm_sw_repo}",
                "sw_repo_user": "${var.ibm_sw_repo_user}"
            },
            "ibm_internal": {
            "roles": "[ihs-wasmode-admin]"
            },
            "ihs": {
                "admin_server": {
                    "enabled": "true",
                    "port": "8008",
                    "username": "${var.referenced-image_ihs_admin_server_username}"
                },
                "install_dir": "/opt/IBM/HTTPServer",
                "install_mode": "admin",
                "plugin": {
                    "enabled": "true",
                    "install_dir": "/opt/IBM/WebSphere/Plugins",
                    "was_webserver_name": "webserver1"
                },
                "port": "${var.referenced-image_ihs_port}",
                "version": "${var.referenced-image_ihs_version}"
            }
        },
        "vault_content": {
            "item": "secrets",
            "values": {
                "ibm": {
                    "im_repo_password": "${var.ibm_im_repo_password}",
                    "sw_repo_password": "${var.ibm_sw_repo_password}"
                },
                "ihs": {
                    "admin_server": {
                        "password": "${var.referenced-image_ihs_admin_server_password}"
                    }
                }
            },
            "vault": "${random_id.stack_id.hex}"
        }
    }
EOT
}
```

Argument Reference

Table 2. Parameters used with the camc_softwaredeploy provider

Type	Name	Description	Schema	Default	Required
Body	access_token	The pattern manager access token that was provided during Content Runtime provisioning	String		Yes
Body	camc_endpoint	The IP address or FQDN of the pattern manager	String		Yes
Body	data	A valid JSON structure	JSON		Yes

Type	Name	Description	Schema	Default	Required
Body	name	A label to aid in identifying the resource	String		No
Data	os_username	The operating system user name used by the pattern manager to remote SSH to the target node	String		Yes
Data	stack_id	The stack ID shared across nodes in this deployment	String		Yes
Data	environment_name	The environment name for this deployment	String		Yes
Data	host_ip	The IP address of the host	String		Yes
Data	node_name	The unique name to give the node on your chef server	String		Yes
Data	node_attributes	Chef attributes to be added to the node object prior to the software deployment process.	String		Yes

CAMC script package

The camc_scriptpackage resource executes scripts and stores the output for use by other resources in your Terraform template. For example, use it to retrieve hostname and IP address information from an IPAM server and use the results in the virtual machine description.

The camc_scriptpackage resource allows you to store scripts locally on the cam-provider-terraform microservice or on an HTTP(s) server, execute scripts or commands locally from the cam-provider-terraform microservice or remotely on a deployed virtual machine, and control when the scripts should be run.

Passing parameters

There are two ways to pass parameters to the script: through the program array and the query JSON object. Parameters specified in the program array are passed to the script as command line parameters. The query JSON object contains name/value pairs of strings and are passed to the script through stdin.

Script response values

Responses from the script are collected and available in the Terraform template. Scripts returning multiple values must return them in a JSON object containing name/value pairs of strings. If the response is not a JSON object, it will be available to the Terraform template in a property called stdout.

Local scripts

Local scripts are run as a non-privileged user within the terraform-provider container in Managed services. This container has a limited number of installed packages, including:

```
awk
base64
bash
curl
gpg
jq
md5sum
Python 2.7.9 - no pip
sed
ssh
tr
wget
xargs
```

If the package that you are looking for does not exist locally, the recommendation is to use the remote script capabilities of the script provider to run scripts on a remote virtual machine that has those packages available.

Example usage

Local script

Example 1

```
resource "camc_scriptpackage" "LocalIp" {
  program = ["/bin/bash", "/var/scripts/get_ip.sh"]
  query = {
    prop1 = "prop1_value"
  }
  on_create = true
}
```

Example 2:

```
resource "camc_scriptpackage" "delete_test_resource" {
depends_on = ["camc_scriptpackage.create_test_resource"]
program = ["/bin/bash", "${path.module}/scripts/delete_ucd_resource.sh", "-h", "${var.ucd_host}", "-u",
"${var.ucd_user}", "-i", "${lookup(camc_scriptpackage.create_test_resource.result, "resource_id")}"]
program_sensitive = ["-p", "${var.ucd_password}"]
on_delete = true
}
```

Local script from HTTPS server

```
resource "camc_scriptpackage" "LocalHttpsScript" {
  program = ["/bin/bash", "/var/scripts/get_ip.sh"]
  source = "https://172.95.34.11/scripts/cam/get_ip.sh"
  source_user = "${var.src_user}"
  source_password = "${var.src_pass}"
  source_no_check_cert = true
  destination = "/var/scripts/get_ip.sh"
  on_create = true
}
```

Remote script

```
resource "camc_scriptpackage" "RemoteScript" {
  program = ["/bin/bash", "/var/scripts/deregister_node.sh"]
  remote_host = "172.13.205.41"
  remote_user = "${var.referenced-image-os_username}"
  remote_password = "${var.referenced-image-os_password}"
  destination = "/var/deregister_node.sh"
  on_delete = true
}
```

Referencing script output

```
output "local_script" {
  value = "${camc_scriptpackage.LocalIp.result}"
}

output "local_script_ip" {
  value = "${lookup(camc_scriptpackage.LocalIp.result, var.ip)}"
}

output "remote_script_stdout" {
  value = "${camc_scriptpackage.CreateScript.result}"
}
```

Argument Reference

Table 3. Parameters used with the camc_scriptpackage provider

Type	Name	Description	Schema	Default	Required
Body	destination	Destination file for the script. This can be the location to download the script from an HTTP server or the remote location to copy your local script	String		Yes, if specifying source
Body	on_create	Execute this script when this resource is created	Boolean	false	No
Body	on_delete	Execute this script when this resource is deleted	Boolean	false	No
Body	on_update	Execute this script when this resource is updated	Boolean	false	No
Body	depends_on	It is a Terraform metadata argument. For more information, see Terraform Resources.	Boolean	false	No
Body	program_sensitive	It is an array that contains a list of sensitive program arguments like passwords or other secret fields. Having them in this list ensures that the value of the attribute is not displayed in logs or regular output. Example: If you want to pass a sensitive field like password to a command in "program" parameter, then you can provide a program_sensitive parameter as `program_sensitive = ["${var.ucd_password}"]`.	Array		No
Body	program	Array containing the command to run	Array		Yes
Body	query	Name/value pairs of parameters passed to your script through stdin	JSON Object		No
Body	remote_host	The hostname or IP address of the remote system to execute this script	String		No

Type	Name	Description	Schema	Default	Required
Body	remote_key	The base64-encoded private key to access the host specified by the remote_host parameter. One of remote_key or remote_password are required when specifying remote_host	String		No
Body	remote_password	The password to access the host specified by the remote_host parameter. One of remote_key or remote_password are required when specifying remote_host	String		No
Body	remote_user	The user to access the host specified by the remote_host parameter	String		Yes, if remote_host is specified
Body	source	The location of the script. This can be a local path on the cam-provider-terraform microservice or an HTTP(s) URL	String		No
Body	source_no_check_cert	Ignore the certificate when downloading the script from the HTTPS server	Boolean	false	No
Body	source_password	The password for the HTTPS server used to host the script	String		Yes, if source is an HTTPS server
Body	source_user	The user for the HTTPS server used to host the script	String		Yes, if source is an HTTPS server
Body	trace	If true, writes more data to TF_LOG	Boolean	False	No

CAMC vault item

Description

IBM Chef-based patterns use the chef-vault gem to store secrets such as passwords and user names to be defined by and used by cookbooks during the software deployment process. The camc_vaultitem resource will specify the items to be created.

It is important to note that nodes are logically arranged into a stack. A stack is a set of nodes belonging to the same pattern and is defined by the stack_id parameter. There is a single chef-vault entry for each stack so that nodes in a stack can share secrets between themselves, but not with other nodes deployed in the environment.

Example Usage

```
resource "camc_vaultitem" "vault1" {
    name = "vault1"
    trace = "true"
    camc_endpoint = "http://${var.endpoint}:5443/v1/vaultitem"
    access_token = "${var.pm_private_b64}"
    data = <<EOT
    {
        "vault_content": {
            "item": "secrets",
            "values": {},
            "vault": "${random_id.stack_id.hex}"
        }
    }
EOT
}
```

Argument Reference

Table 4. Parameters used with the camc_vaultitem provider

Type	Name	Description	Schema	Default	Required
Body	access_token	The pattern manager access token that was provided during Content Runtime provisioning	String		Yes
Body	camc_endpoint	The IP address or FQDN of the pattern manager	String		Yes
Body	data	A valid JSON structure	JSON		Yes
Body	name	A label to aid in identifying the resource	String		No
Data	vault_content	The content to add to the vault	String		Yes
Body	trace	If true, writes more data to TF_LOG	Boolean	False	No

Shipped Terraform providers

The page lists all the Terraform providers and their versions that are shipped for each platform.

- Terraform providers in 3.4.x or terraform version less than 0.13.0
- Terraform providers in 3.4.x for terraform version greater than 0.13.0

Terraform providers in 3.4.x or terraform version less than 0.13.0

Intel providers	IBM Power providers
terraform-provider-alicloud_v1.211.0	terraform-provider-alicloud_v1.211.0
terraform-provider-aws_v5.19.0	terraform-provider-archive_v2.4.0
terraform-provider-azurerm_v3.75.0	terraform-provider-arukas_v1.1.0
terraform-provider-camc_v0.3.0	terraform-provider-aws_v5.19.0
terraform-provider-external_v2.3.1	terraform-provider-azurerm_v3.75.0
terraform-provider-google_v5.0.0	terraform-provider-bitbucket_v1.2.0
terraform-provider-http_v3.4.0	terraform-provider-camc_v0.3.0
terraform-provider-ibm_v1.58.1	terraform-provider-chef_v0.2.0
terraform-provider-kubernetes_v2.23.0	terraform-provider-circonus_v0.12.15
terraform-provider-local_v2.4.0	terraform-provider-clc_v0.1.0
terraform-provider-nsxt_v3.3.2	terraform-provider-cloudflare_v4.16.0
terraform-provider-null_v3.2.1	terraform-provider-cloudstack_v0.3.0
terraform-provider-nutanix_v1.1.1	terraform-provider-cobbler_v1.1.0
terraform-provider-openstack_v1.52.1	terraform-provider-consul_v2.18.0
terraform-provider-random_v3.5.1	terraform-provider-datadog_v2.12.0
terraform-provider-tencentcloud_v1.81.32	terraform-provider-digitalocean_v2.30.0
terraform-provider-ucd_v0.6.3	terraform-provider-dme_v0.1.1
terraform-provider-tls_v4.0.4	terraform-provider-dns_v3.3.2
terraform-provider-vra7_v3.0.9	terraform-provider-dnsimple_v0.4.0
terraform-provider-vsphere_v2.4.3	terraform-provider-docker_v2.7.2
terraform-provisioner-ucd_v0.6.3	terraform-provider-dyn_v1.2.0
	terraform-provider-external_v2.3.1
	terraform-provider-fastly_v0.19.2
	terraform-provider-github_v4.1.0
	terraform-provider-gitlab_v2.11.0
	terraform-provider-google_v5.0.0
	terraform-provider-grafana_v1.5.0
	terraform-provider-heroku_v2.5.0
	terraform-provider-http_v3.4.0
	terraform-provider-ibm_v1.58.1
	terraform-provider-icinga2_v0.4.0
	terraform-provider-ignition_v1.2.1
	terraform-provider-influxdb_v1.3.1
	terraform-provider-kubernetes_v2.23.0
	terraform-provider-librato_v0.1.0
	terraform-provider-local_v2.4.0
	terraform-provider-logentries_v1.0.0
	terraform-provider-logicmonitor_v2.0.10
	terraform-provider-mailgun_v0.4.3
	terraform-provider-mysql_v1.9.0
	terraform-provider-newrelic_v3.27.1
	terraform-provider-nomad_v2.0.0
	terraform-provider-ns1_v2.0.8
	terraform-provider-nsxt_v3.3.2
	terraform-provider-null_v3.2.1
	terraform-provider-nutanix_v1.1.0
	terraform-provider-oneandone_v1.3.0
	terraform-provider-opc_v1.4.1
	terraform-provider-openstack_v1.52.1
	terraform-provider-opsgenie_v0.6.30
	terraform-provider-ovh_v0.9.0

Intel providers	IBM Power providers
	terraform-provider-packet_v3.2.1
	terraform-provider-pagerduty_v3.0.1
	terraform-provider-postgresql_v1.7.1
	terraform-provider-powerdns_v1.4.0
	terraform-provider-profitbricks_v1.5.7
	terraform-provider-rabbitmq_v1.4.0
	terraform-provider-rancher_v1.5.0
	terraform-provider-random_v3.5.1
	terraform-provider-rundeck_v0.4.0
	terraform-provider-scaleway_v2.28.1
	terraform-provider-softlayer_v0.0.1
	terraform-provider-spotinst_v1.25.0
	terraform-provider-statuscake_v1.0.0
	terraform-provider-template_v2.2.0
	terraform-provider-tencentcloud_v1.18.32
	terraform-provider-terraform_v1.0.2
	terraform-provider-tls_v4.0.4
	terraform-provider-triton_v0.8.2
	terraform-provider-ucd_v0.6.3
	terraform-provider-ultradns_v0.2.0
	terraform-provider-vault_v3.6.1
	terraform-provider-vcd_v2.9.0
	terraform-provider-vra7_v3.0.9
	terraform-provider-vsphere_v2.4.3
	terraform-provisioner-ucd_v0.6.3

Terraform providers in 3.4.x for terraform version greater than 0.13.0

Intel providers	IBM Power providers
registry.ibm.com/cam/camc/0.3.0/linux_amd64/terraform-provider-camc_v0.3.0	registry.ibm.com/cam/camc/0.3.0/linux_amd64/terraform-provider-camc_v0.3.0
registry.ibm.com/cam/ucd/0.6.3/linux_amd64/terraform-provider-ucd_v0.6.3	registry.ibm.com/cam/ucd/0.6.3/linux_amd64/terraform-provider-ucd_v0.6.3
registry.ibm.com/cam/ucd/0.6.3/linux_amd64/terraform-provider-ucd_v0.6.3	registry.ibm.com/cam/ucd/0.6.3/linux_amd64/terraform-provider-ucd_v0.6.3
registry.terraform.io/aliyun/alicloud/1.211.0/linux_amd64/terraform-provider-alicloud_v1.211.0	registry.terraform.io/aliyun/alicloud/1.211.0/linux_ppc64le/terraform-provider-alicloud_v1.211.0
registry.terraform.io/hashicorp/aws/5.19.0/linux_amd64/terraform-provider-aws_v5.19.0	registry.terraform.io/circonus-labs/circonus/0.12.5/linux_ppc64le/terraform-provider-circonus_v0.12.5
registry.terraform.io/hashicorp/azurerm/3.75.0/linux_amd64/terraform-provider-azurerm_v3.75.0	registry.terraform.io/cloudflare/cloudflare/4.16.0/linux_ppc64le/terraform-provider-cloudflare_v4.16.0
registry.terraform.io/hashicorp/external/2.3.1/linux_amd64/terraform-provider-external_v2.3.1	registry.terraform.io/digitalocean/digitalocean/2.3.0/linux_ppc64le/terraform-provider-digitalocean_v2.3.0
registry.terraform.io/hashicorp/google/5.0.0/linux_amd64/terraform-provider-google_v5.0.0	registry.terraform.io/hashicorp/archive/2.4.0/linux_ppc64le/terraform-provider-archive_v2.4.0
registry.terraform.io/hashicorp/http/3.4.0/linux_amd64/terraform-provider-http_v3.4.0	registry.terraform.io/hashicorp/arukas/1.1.0/linux_ppc64le/terraform-provider-arukas_v1.1.0
registry.terraform.io/hashicorp/kubernetes/2.23.0/linux_amd64/terraform-provider-kubernetes_v2.23.0	registry.terraform.io/hashicorp/aws/5.19.0/linux_ppc64le/terraform-provider-aws_v5.19.0
registry.terraform.io/hashicorp/local/2.4.0/linux_amd64/terraform-provider-local_v2.4.0	registry.terraform.io/hashicorp/azurerm/3.75.0/linux_ppc64le/terraform-provider-azurerm_v3.75.0
registry.terraform.io/hashicorp/null/3.2.1/linux_amd64/terraform-provider-null_v3.2.1	registry.terraform.io/hashicorp/bitbucket/1.2.0/linux_ppc64le/terraform-provider-bitbucket_v1.2.0
registry.terraform.io/hashicorp/nutanix/1.1.0/linux_amd64/terraform-provider-nutanix_v1.1.0	registry.terraform.io/hashicorp/chef/0.2.0/linux_ppc64le/terraform-provider-chef_v0.2.0
registry.terraform.io/hashicorp/random/3.5.1/linux_amd64/terraform-provider-random_v3.5.1	registry.terraform.io/hashicorp/clc/0.1.0/linux_ppc64le/terraform-provider-clc_v0.1.0

Intel providers	IBM Power providers
registry.terraform.io/hashicorp/tls/4.0.4/linux_amd64/terraform-provider-tls_v4.0.4	registry.terraform.io/hashicorp/cobbler/1.1.0/linux_ppc64le/terraform-provider-cobbler_v1.1.0
registry.terraform.io/hashicorp/vsphere/2.1.1/linux_amd64/terraform-provider-vsphere_v2.4.3	registry.terraform.io/hashicorp/consul/2.18.0/linux_ppc64le/terraform-provider-consul_v2.18.0
registry.terraform.io/IBM-Cloud/ibm/1.58.1/linux_amd64/terraform-provider-ibm_v1.58.1	registry.terraform.io/hashicorp/datadog/2.12.0/linux_ppc64le/terraform-provider-datadog_v2.12.0
registry.terraform.io/tencentcloudstack/tencentcloud/1.72.5/linux_amd64/terraform-provider-tencentcloud_v1.81.32	registry.terraform.io/hashicorp/dme/0.1.1/linux_ppc64le/terraform-provider-dme_v0.1.1
registry.terraform.io/terraform-provider-openstack/openstack/1.52.1/linux_amd64/terraform-provider-openstack_v1.52.1	registry.terraform.io/hashicorp/dns/3.3.2/linux_ppc64le/terraform-provider-dns_v3.3.2
registry.terraform.io/vmware/nsxt/3.3.2/linux_amd64/terraform-provider-nsxt_v3.3.2	registry.terraform.io/hashicorp/dnsimple/0.4.0/linux_ppc64le/terraform-provider-dnsimple_v0.4.0
registry.terraform.io/vmware/vra7/3.0.9/linux_amd64/terraform-provider-vra7_v3.0.9	registry.terraform.io/hashicorp/docker/2.7.2/linux_ppc64le/terraform-provider-docker_v2.7.2
	registry.terraform.io/hashicorp/dyn/1.2.0/linux_ppc64le/terraform-provider-dyn_v1.2.0
	registry.terraform.io/hashicorp/external/2.3.1/linux_ppc64le/terraform-provider-external_v2.3.1
	registry.terraform.io/hashicorp/fastly/0.19.2/linux_ppc64le/terraform-provider-fastly_v0.19.2
	registry.terraform.io/hashicorp/github/4.1.0/linux_ppc64le/terraform-provider-github_v4.1.0
	registry.terraform.io/hashicorp/gitlab/2.11.0/linux_ppc64le/terraform-provider-gitlab_v2.11.0
	registry.terraform.io/hashicorp/google/5.0.0/linux_ppc64le/terraform-provider-google_v5.0.0
	registry.terraform.io/hashicorp/grafana/1.5.0/linux_ppc64le/terraform-provider-grafana_v1.5.0
	registry.terraform.io/hashicorp/heroku/2.5.0/linux_ppc64le/terraform-provider-heroku_v2.5.0
	registry.terraform.io/hashicorp/http/3.4.0/linux_ppc64le/terraform-provider-http_v3.4.0
	registry.terraform.io/hashicorp/icinga2/0.4.0/linux_ppc64le/terraform-provider-icinga2_v0.4.0
	registry.terraform.io/hashicorp/ignition/1.2.1/linux_ppc64le/terraform-provider-ignition_v1.2.1
	registry.terraform.io/hashicorp/influxdb/1.3.1/linux_ppc64le/terraform-provider-influxdb_v1.3.1
	registry.terraform.io/hashicorp/kubernetes/2.23.0/linux_ppc64le/terraform-provider-kubernetes_v2.23.0
	registry.terraform.io/hashicorp/librato/0.1.0/linux_ppc64le/terraform-provider-librato_v0.1.0
	registry.terraform.io/hashicorp/local/2.4.0/linux_ppc64le/terraform-provider-local_v2.4.0
	registry.terraform.io/hashicorp/logentries/1.0.0/linux_ppc64le/terraform-provider-logentries_v1.0.0
	registry.terraform.io/hashicorp/mailgun/0.4.3/linux_ppc64le/terraform-provider-mailgun_v0.4.3
	registry.terraform.io/hashicorp/mysql/1.9.0/linux_ppc64le/terraform-provider-mysql_v1.9.0
	registry.terraform.io/hashicorp/nomad/2.0.0/linux_ppc64le/terraform-provider-nomad_v2.0.0
	registry.terraform.io/hashicorp/null/3.2.1/linux_ppc64le/terraform-provider-null_v3.2.1
	registry.terraform.io/hashicorp/nutanix/1.1.0/linux_ppc64le/terraform-provider-nutanix_v1.1.0
	registry.terraform.io/hashicorp/oneandone/1.3.0/linux_ppc64le/terraform-provider-oneandone_v1.3.0
	registry.terraform.io/hashicorp/opc/1.4.1/linux_ppc64le/terraform-provider-opc_v1.4.1

Intel providers	IBM Power providers
	registry.terraform.io/hashicorp/ovh/0.9.0/linux_ppc64le/terraform-provider-ovh_v0.9.0
	registry.terraform.io/hashicorp/postgresql/1.7.1/linux_ppc64le/terraform-provider-postgresql_v1.7.1
	registry.terraform.io/hashicorp/powerdns/1.4.0/linux_ppc64le/terraform-provider-powerdns_v1.4.0
	registry.terraform.io/hashicorp/profitbricks/1.5.7/linux_ppc64le/terraform-provider-profitbricks_v1.5.7
	registry.terraform.io/hashicorp/rabbitmq/1.4.0/linux_ppc64le/terraform-provider-rabbitmq_v1.4.0
	registry.terraform.io/hashicorp/rancher/1.5.0/linux_ppc64le/terraform-provider-rancher_v1.5.0
	registry.terraform.io/hashicorp/random/3.5.1/linux_ppc64le/terraform-provider-random_v3.5.1
	registry.terraform.io/hashicorp/rundeck/0.4.0/linux_ppc64le/terraform-provider-rundeck_v0.4.0
	registry.terraform.io/hashicorp/softlayer/0.0.1/linux_ppc64le/terraform-provider-softlayer_v0.0.1
	registry.terraform.io/hashicorp/spotinst/1.25.0/linux_ppc64le/terraform-provider-spotinst_v1.25.0
	registry.terraform.io/hashicorp/statuscake/1.0.0/linux_ppc64le/terraform-provider-statuscake_v1.0.0
	registry.terraform.io/hashicorp/template/2.2.0/linux_ppc64le/terraform-provider-template_v2.2.0
	registry.terraform.io/hashicorp/terraform/1.0.2/linux_ppc64le/terraform-provider-terraform_v1.0.2
	registry.terraform.io/hashicorp/tls/4.0.4/linux_ppc64le/terraform-provider-tls_v4.0.4
	registry.terraform.io/hashicorp/ultradns/0.2.0/linux_ppc64le/terraform-provider-ultradns_v0.2.0
	registry.terraform.io/hashicorp/vault/3.20.1/linux_ppc64le/terraform-provider-vault_v3.20.1
	registry.terraform.io/hashicorp/vcd/2.9.0/linux_ppc64le/terraform-provider-vcd_v2.9.0
	registry.terraform.io/hashicorp/vsphere/2.4.3/linux_ppc64le/terraform-provider-vsphere_v2.4.3
	registry.terraform.io/IBM-Cloud/ibm/1.58.1/linux_ppc64le/terraform-provider-ibm_v1.58.1
	registry.terraform.io/logicmonitor/logicmonitor/2.0.10/linux_ppc64le/terraform-provider-logicmonitor_v2.0.10
	registry.terraform.io/newrelic/newrelic/3.27.1/linux_ppc64le/terraform-provider-newrelic_v3.27.1
	registry.terraform.io/ns1-terraform/ns1/2.0.8/linux_ppc64le/terraform-provider-ns1_v2.0.8
	registry.terraform.io/opsgenie/opsgenie/0.6.30/linux_ppc64le/terraform-provider-opsgenie_v0.6.30
	registry.terraform.io/packethost/packet/3.2.1/linux_ppc64le/terraform-provider-packet_v3.2.1
	registry.terraform.io/PagerDuty/pagerduty/3.0.1/linux_ppc64le/terraform-provider-pagerduty_v3.0.1
	registry.terraform.io/scaleway/scaleway/2.28.0/linux_ppc64le/terraform-provider-scaleway_v2.28.0
	registry.terraform.io/tencentcloudstack/tencentcloud/1.81.32/linux_ppc64le/terraform-provider-tencentcloud_v1.81.32
	registry.terraform.io/terraform-provider-openstack/openstack/1.52.1/linux_ppc64le/terraform-provider-openstack_v1.52.1
	registry.terraform.io/TritonDataCenter/triton/0.8.2/linux_ppc64le/terraform-provider-triton_v0.8.2
	registry.terraform.io/vmware/nsxt/3.3.2/linux_ppc64le/terraform-provider-nsxt_v3.3.2

Intel providers	IBM Power providers
	registry.terraform.io/vmware/vra7/3.0.9/linux_ppc64le/terraform-provider-vra7_v3.0.9
	registry.terraform.io/xanzy/cloudstack/0.3.0/linux_ppc64le/terraform-provider-cloudstack_v0.3.0

Supported Terraform Templates

The page lists all the Terraform templates that are supported by IBM and pre-loaded into the product. Rest of the templates are provided as-is and user can import them into the product using these Import instructions or fork the repository and change the template to suite their needs and use it.

- Starterlibrary Terraform Templates
- Starterlibrary Cluster Creation Terraform Templates
- Integration Templates
- Middleware Templates

These templates can be found in following GitHub

Starterlibrary Terraform Templates

Repository Location	Template Name
starterlibrary/AWS/terraform1XX/hcl/lamp/	LAMP stack deployment on AWS for Terraform 1.x.x
starterlibrary/AWS/terraform1XX/hcl/meanstack/	MEAN stack deployment on AWS for Terraform 1.x.x
starterlibrary/AWS/terraform1XX/hcl/rdssnap/	Amazon Relational Database Service Snapshot for Terraform 1.x.x
starterlibrary/AWS/terraform1XX/hcl/storage/	Amazon S3 Storage for Terraform 1.x.x
starterlibrary/AWS/terraform1XX/hcl/rds/	Amazon Relational Database Service for Terraform 1.x.x
starterlibrary/AWS/terraform1XX/hcl/lampinfra/	LAMP infrastructure deployment on AWS for Terraform 1.x.x
starterlibrary/AWS/terraform1XX/singlefile/hcl/	Amazon EC2 Virtual Server with SSH key for Terraform 1.x.x
starterlibrary/Google/terraform1XX/hcl/singleVM/	Google Cloud Single Virtual Machine Example for Terraform 1.x.x
starterlibrary/Google/terraform1XX/hcl/lamp/	Google Cloud LAMP stack deployment for Terraform 1.x.x
starterlibrary/BlueMix/terraform1XX/hcl/scenario1/	Virtual Server with SSH key for Terraform 1.x.x
starterlibrary/BlueMix/terraform1XX/hcl/scenario2/	2 Virtual Servers with SSH keys for Terraform 1.x.x
starterlibrary/BlueMix/terraform1XX/hcl/lamp-hybrid/	LAMP stack deployment (hybrid-cloud) for Terraform 1.x.x
starterlibrary/BlueMix/terraform1XX/hcl/meanstack-hybrid/	MEAN stack deployment (hybrid-cloud) for Terraform 1.x.x
starterlibrary/BlueMix/terraform1XX/hcl/scenario3/	2 Virtual Servers with SSH keys with one on a private network for Terraform 1.x.x
starterlibrary/VRA/terraform1XX/hcl/single-catalog-vm-deploy/	VMware vRealize Automation single node catalog deployment for Terraform 1.x.x
starterlibrary/OpenStack/terraform1XX/hcl/single-vm-deploy/	OpenStack Single Virtual Machine for Terraform 1.x.x
starterlibrary/Tencent/terraform1XX/hcl/singleVM/	Tencent cloud Virtual Server with SSH key for Terraform 1.x.x
starterlibrary/VMwareNSX-T/terraform1XX/hcl/switch/	VMware NSX-T Sample to create a logical switch for Terraform 1.x.x
starterlibrary/Other/Approval-1XX/InfrastructureManagement/	Poll Infrastructure Management for Approval for Terraform 1.x.x
starterlibrary/IBM_Cloud/database-service-1XX/	IBM Database Service for Terraform 1.x.x
starterlibrary/Alicloud/terraform1XX/hcl/singleVM/	Alibaba cloud ECS Virtual Server with SSH key for Terraform 1.x.x
starterlibrary/VMware/terraform1XX/hcl/singleVM/	SingleVirtualMachine for Terraform 1.x.x
starterlibrary/VMware/terraform1XX/hcl/lamp/	LAMP stack deployment on VMware for Terraform 1.x.x
starterlibrary/VMware/terraform1XX/hcl/singleWinVM/	SingleWindowsVirtualMachine for Terraform 1.x.x
starterlibrary/VMware/terraform1XX/hcl/meanstack/	MEAN stack deployment on VMware for Terraform 1.x.x
starterlibrary/VMware/terraform1XX/hcl/singleVMWithPrivateSSHKey/	Single Virtual Machine with Private SSH Key for Terraform 1.x.x
starterlibrary/Azure/terraform1XX/hcl/lamp/	LAMP stack deployment on Microsoft Azure for Terraform 1.x.x
starterlibrary/Azure/terraform1XX/hcl/meanstack/	MEAN stack deployment on Microsoft Azure for Terraform 1.x.x
starterlibrary/Azure/terraform1XX/hcl/singlevirtualmachine/	Virtual Machine with SSH key on Microsoft Azure for Terraform 1.x.x

Starterlibrary Cluster Creation Terraform Templates

Repository Location	Template Name

Repository Location	Template Name
template_ibm_roks/terraform/	Red Hat OpenShift Single Zone Cluster on IBM Cloud for Terraform 1.x.x
template_kubernetes_iks/terraform1XX/	IBM Cloud Kubernetes Service for Terraform 1.x.x
template_kubernetes_eks/terraform1XX/	Amazon Web Services EKS for Terraform 1.x.x
template_kubernetes_aks/terraform1XX/	Microsoft Azure Kubernetes Service for Terraform 1.x.x
template_kubernetes_gke/terraform1XX/	Google Kubernetes Engine for Terraform 1.x.x

Integration Templates

Repository Location	Template Name
template_integration_servicenow/other/terraform1XX/	ServiceNow CMDB Configuration Template for Terraform 1.x.x

IBM Cloud Pak for AIOps troubleshooting

Learn how to isolate and resolve problems with IBM Cloud Pak® for AIOps.

Verify that your issues are not related to operating system requirements, such as disk, memory, and CPU capacities. For more information, see Planning.

Additionally, you can see whether the problem is documented in Known issues and limitations.

If you need direct help from IBM Support, see the Support.

Troubleshooting topics

- Install and upgrade
- Clusters and nodes
- Connections
- AI model management
- ChatOps
- Topology
- User management (IAM)
- Secure Tunnel
- Alert management
- Runbook Automation
- Infrastructure Automation

Troubleshooting installation and upgrade

Learn how to isolate and resolve problems with installing and upgrading IBM Cloud Pak for AIOps.

Known issues and limitations

In addition to the common problems and solutions that are listed, review the installation related known issues for potential issues that you might encounter during installation. For more information, see Known issues and limitations.

General troubleshooting issues

- General troubleshooting tips (Console)
- General troubleshooting tips (CLI)
- Troubleshooting storage
- Recovering IBM Cloud Pak for AIOps after a cluster restart

Note: If you have an aborted or stalled installation, uninstall any installed or partially installed components and restart the installation. For more information about uninstalling, see Uninstalling.

Troubleshooting installation issues

- Installation fails to proceed when creating the IBM Cloud Pak for AIOps custom resource

- The AI manager operator does not progress past `Reconciling`, and the Cassandra setup pod is stuck
- LifecycleService shows as 'Not Ready'
- Offline install or upgrade stuck because the 'oc ibm-pak generate mirror-manifests' command fails with 'no space left on device'
- "Application is not available" message is displayed intermittently
- After certificate renewal the Kafka listener status type is not populated, or <CP4AIOPS_INSTALLATION_NAME>-kafka-secret is missing

Troubleshooting upgrade issues

- Custom roles with deprecated permissions after upgrade
- Pods in "Pending" state after upgrade
- Upgrade does not complete successfully, with error *Cannot delete AutomationBase instance*
- After upgrade, policies do not process alerts due to problem with the `aiops-ir-lifecycle-create-policies-job`
- Upgrade to 4.2.1 hangs, with IssueResolutionCore stuck in the "Update in progress" state
- Metric anomaly alerts are no longer generated after upgrade
- After upgrade, closed alerts are not cleared from the alert list
- Vault operator throws constraints not satisfiable error, and upgrade stalls
- New alerts from metric anomaly detection are not showing
- After upgrade, Netcool and Metrics connections have a red status in the Connections UI
- Anomalies from log connections are not being detected
- Offline install or upgrade stuck because the 'oc ibm-pak generate mirror-manifests' command fails with 'no space left on device'
- After upgrading Red Hat OpenShift, AIOpsEdge is in an unhealthy state with a certification verification error

General troubleshooting

General troubleshooting tips (Console)

1. From the Red Hat® OpenShift® console, go to **Workloads** > **Pods**.
2. Select the project that IBM Cloud Pak for AIOps is installed in from the drop-down menu to view the status of all the pods in this project.
3. If you want to see the status and details for one of the pods, click the pod that you want to investigate.
4. If you want to see the logs for the pod, click **Logs**.

General troubleshooting tips (CLI)

- To find out the status of all the pods in a project, run the following command:

  ```
  oc get pods -n <project>
  ```

  Where `<project>` is the project (namespace) where IBM Cloud Pak for AIOps is deployed.

- To see the status of a particular pod, run the following command:

  ```
  oc get pods | grep <pod_name>
  ```

  Where `<pod_name>` is the pod that you want to see status for.

- To see a pod's logs, run the following command:

  ```
  oc logs -f <pod_name> -n <project>
  ```

  Where

  - `<pod_name>` is the pod that you want to see status for.
  - `<project>` is the project (namespace) where IBM Cloud Pak for AIOps is deployed.

Troubleshooting storage

Storage that is associated with persistent volumes claims (PVCs) reaches capacity. This issue can occur as storage requests and resources fluctuate over time.

Solution: If storage fills up, increase the size of the PVC. For more information about increasing the size of a PVC, see Storage class requirements.

Recovering IBM Cloud Pak for AIOps after a cluster restart

After a Red Hat® OpenShift® Container Platform cluster restart, your IBM Cloud Pak for AIOps deployment has not come up successfully.

Use the following steps to help you to identify any problems.

1. Run the following command and then check that all of the OpenShift nodes come back up, and that they all report a *STATUS* of `Ready`.

```
oc get nodes
```

If all of the OpenShift nodes have not come back up, then see the *Updating clusters* and *Troubleshooting* sections in the OpenShift documentation for guidance.

2. Run the following command and then check that the IBM Cloud Pak foundational services pods all have a *STATUS* OF **Running** or **Completed**.

```
oc get pods -n ibm-common-services
```

3. Run the following command and then check that the IBM Cloud Pak for AIOps pods all have a *STATUS* OF **Running** or **Completed**.

```
oc get pods -n <project>
```

Where **<project>** is the project (namespace) where IBM Cloud Pak for AIOps is deployed.

4. Check the logs of any pods that did not report a *STATUS* of **Running** or **Completed** in steps 2 and 3 to see whether you can identify a cause.

```
oc logs -f <failed_pod_name> -n <project>
```

Where

- **<project>** is the project (namespace) where IBM Cloud Pak for AIOps is deployed.
- **<failed_pod_name>** is a failed pod name, as returned in steps 2 and 3.

Solution: Restart any failed pods with the following command:

```
oc delete pod <failed_pod_name> -n <project>
```

Where

- **<project>** is the project (namespace) where IBM Cloud Pak for AIOps is deployed.
- **<failed_pod_name>** is the pod name as returned in steps 2 and 3.

Troubleshooting installation

Installation not proceeding after creating the IBM Cloud Pak for AIOps custom resource

When you are installing, the installation is not proceeding as expected after you create an instance of the IBM Cloud Pak for AIOps custom resource. For instance, after you create the instance, the installation process appears to be stalled, however no pods in the namespace and cluster display show errors.

This stalling of the installation process can be due to an issue with the Operator Lifecycle Manager (OLM), which causes the expected ClusterServiceVersion (CSV) for some operator subscriptions to not be found. For example, installPlan instances for the **aimanager-operator**, **connector-utilities-operator**, and other operators can be found, but without a reference to the required ClusterServiceVersion (CSV) YAML. While the operators can be created within your defined namespace, if the ClusterServiceVersion (CSV) cannot be found, the operators cannot be managed. If this issue occurs, the overall installation cannot proceed correctly.

Solution: Delete any operator subscription that does not have an associated ClusterServiceVersion (CSV) YAML and has an unresolved subscription. The installation process should re-create the associated operator and generate a new installPlan. If a new installPlan is not generated, repeat the process to delete the operator subscription and try again.

The AI manager operator does not progress past `Reconciling`, and the Cassandra setup pod is stuck

When installing, the AI Manager operator remains in **Phase: Reconciling**. The **aiops-ir-analytics-cassandra-setup** pod is in Crashloopbackoff state, and its log has the following error:

```
ERROR [2023-06-13 09:57:44,010] com.ibm.itsm.analytics.metric.service.cassandra.SessionManager: Cassandra not available.
```

Solution: Use the following steps to reset the Cassandra admin password, and delete the **aiops-ir-analytics-cassandra-setup** and **aiops-ir-core-archiving-setup** jobs.

1. Run the following command to switch to your IBM Cloud Pak for AIOps project.

```
oc project <project>
```

Where **<project>** is the project that your IBM Cloud Pak for AIOps installation is deploying in.

2. Reset the Cassandra admin password.

```
oc rsh aiops-topology-cassandra-0 bash -c 'python3 -u /opt/ibm/change-superuser.pyc --initial --user admin --password $(cat /opt/ibm/cassandra_auth/password) --replication-factor
```

```
$CASSANDRA_AUTH_REPLICATION_FACTOR'
```

3. Delete the **aiops-ir-analytics-cassandra-setup** job, if it does not own any pods.
    1. Check if there are any pods belonging to the **aiops-ir-analytics-cassandra-setup** job.

        ```
        oc get pod -l app.kubernetes.io/managed-by=aiops-analytics-
        operator,app.kubernetes.io/component=cassandra-setup
        ```

    2. If no pods are returned, then delete the **aiops-ir-analytics-cassandra-setup** job.

        ```
        oc delete job -l app.kubernetes.io/managed-by=aiops-analytics-
        operator,app.kubernetes.io/component=cassandra-setup
        ```

4. Delete the **aiops-ir-core-archiving-setup** job, if it does not own any pods.
    1. Check if there are any pods belonging to the **aiops-ir-core-archiving-setup** job.

        ```
        oc get pod -l app.kubernetes.io/managed-by=ir-core-operator,app.kubernetes.io/component=archiving-
        setup
        ```

    2. If no pods are returned, then delete the **aiops-ir-core-archiving-setup** job.

        ```
        oc delete job -l app.kubernetes.io/managed-by=ir-core-
        operator,app.kubernetes.io/component=archiving-setup
        ```

LifecycleService shows as 'Not Ready'

The job **aiops-ir-lifecycle-create-policies-job** does not complete successfully, and the **aiops-ir-lifecycle-policy-registry-svc-job** log has entries similar to the following:

```
ResponseError: User admin has no CREATE permission on <keyspace aiops_policies> or any of its parents at
FrameReader.readError (/app/node_modules/cassandra-driver/lib/readers.js:389:17)
```

Solution: Restart the policy jobs by running the following commands:

```
oc delete job aiops-ir-lifecycle-policy-registry-svc-job
oc delete job aiops-ir-lifecycle-create-policies-job
```

"Application is not available" message is displayed intermittently.

This is caused by a defect in some versions of Red Hat OpenShift. For more information, see OpenShift 4 - Intermittent 503 from all routes or timeouts when calling to backends directly when **allow-from-openshift-ingress** network policy is applied in the Red Hat OpenShift documentation.

Solution: Upgrade to a supported version of Red Hat OpenShift. For more information, see Supported Red Hat OpenShift Container Platform versions.

After certificate renewal the Kafka listener status type is not populated, or <CP4AIOPS_INSTALLATION_NAME>-kafka-secret is missing

On online deployments, the Kafka listener status type is not populated. Run the following command to check the Kafka status:

```
export AIOPS_NAMESPACE=<AIOps Namespace>
oc get kafkas.ibmevents.ibm.com iaf-system -n "${AIOPS_NAMESPACE}" -o jsonpath='{.status.listeners[0].type}'
```

If the command returns no output, then use one of the options in the **Solution** section to fix it.

Solution:

Option A:

Upgrade to IBM Cloud Pak for AIOps 4.5.1 or later.

Option B:

Fix the Events Operator in your IBM Cloud Pak for AIOps deployment.

1. Run the following command to find your Events Operator version:

    ```
    oc get subscriptions.operators.coreos.com ibm-events-operator -o jsonpath='{.status.installedCSV}' -n
    "${AIOPS_NAMESPACE}"
    ```

    If the returned text ends with "v5.0.1", then Events Operator v5.0.1 is installed. If the returned text ends with "v5.0.0", then Events Operator v5.0.0 is installed.

2. If Events Operator 5.0.0 is installed, upgrade the Events Operator to v5.0.1. Upgrade occurs automatically if automatic polling is enabled on the cluster's CatalogSource, and the Subscription is set to automatic approval mode. Contact the IBM Cloud Pak for AIOps team if you have any questions.

3. Run the following command:

```
oc patch kafkas.ibmevents.ibm.com iaf-system -n "${AIOPS_NAMESPACE}" --type=json -
p='[{"op":"remove","path":"/spec/kafka/listeners/0"}]'
```

4. After ten minutes, run the following command:

```
oc get kafkas.ibmevents.ibm.com iaf-system -n "${AIOPS_NAMESPACE}" -o
jsonpath='{.status.listeners[0].type}'
```

   If the command returns a string value, typically **tls**, then the fix is successful. If nothing is returned after an hour, then contact IBM Support.

Troubleshooting upgrade

Custom roles with deprecated permissions after upgrade

When you upgrade IBM Cloud Pak for AIOps, any custom roles that you created which now have deprecated permissions show an invalid permissions error on the **Access Control** page, with the following format:

```
The test custom role 320test role includes invalid permissions: waiops_view_all_alerts,
waiops_view_all_stories. A developer must use the platform APIs to update the test custom role 320test role
to remove the waiops_view_all_alerts, waiops_view_all_stories keys from the permissions array. Specifically,
use the PUT /v1/role/test%20custom%20role%20320test command to edit the role.
```

This issue can occur as the upgraded IBM Cloud Pak® foundational services has deprecated the permissions that are used by the custom role.

Solution: Edit and update the affected roles with the currently available permissions, or delete and recreate the affected roles. For more information, see Roles and permissions.

Pods in "Pending" state after upgrade

On upgrading from IBM Cloud Pak® for AIOps 3.7.2 to 4.1.0 or higher, you might encounter an issue where the following two pods are in **Pending** state after the upgrade:

- **aiops-ir-analytics-metric-spark**
- **aiops-ir-analytics-spark-worker**

Solution: To resolve this issue, complete the following steps:

1. Run the following command to switch to your IBM Cloud Pak® for AIOps project:

```
  oc project <project>
```

   Where **<project>** is the project that your Cloud Pak for AIOps installation is deploying in.

2. If the **oc get pvc aiops-ir-analytics-spark-shared-state** command returns the following output:

```
# oc get pvc
NAME                                                                STATUS    VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS       AGE
aiops-ir-analytics-spark-shared-state                               Pending
my-storage-class   6h44m
```

3. Verify that there are some pods in pending state:

```
aiops-ir-analytics-metric-spark-677d4cdf97-6mqm5                    0/1     Pending     0
59m
aiops-ir-analytics-spark-worker-1                                   0/1     Pending     0
64m
```

4. Correct the storage class value in the aiops AIOpsAnalyticsOrchestrator CR with the following command:

```
oc patch aiopsanalyticsorchestrators.ai.ir.aiops.ibm.com aiops --type=merge -p '{"spec":
{"sparkWorkerPersistentVolumeSpec":{"storageClass":"'$(oc get installations.orchestrator.aiops.ibm.com -
o jsonpath='{.items[0].spec.storageClass}')'"}}}'
```

5. Run the following command to delete the PVC, it recreates itself:

```
oc delete pvc aiops-ir-analytics-spark-shared-state --force --grace-period=0
```

6. The PVC is recreated and should look similar to the following example:

```
# oc get pvc aiops-ir-analytics-spark-shared-state
NAME                                      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS              AGE
aiops-ir-analytics-spark-shared-state   Bound    pvc-8de24327-8b3f-4101-b5cd-f27e7765e996   100Gi
RWX            ocs-storagecluster-cephfs   10d
```

Upgrade does not complete successfully, with error *Cannot delete AutomationBase instance*

IBM® Automation foundation was removed in IBM Cloud Pak for AIOps 4.1.0. Upgrade does not progress because there are no subscriptions for the IBM Automation foundation operators, but their resources are deployed on the cluster and are not cleaned up.

The **ibm-aiops-orchestrator-controller-manager** pod log has an error similar to the following:

```
2023-06-25T03:05:33.440Z        ERROR   Reconciler error        {"controller": "installation",
"controllerGroup": "orchestrator.aiops.ibm.com", "controllerKind": "Installation", "installation":
{"name":"ibm-cp-aiops","namespace":"cp4aiops"}, "namespace": "cp4aiops", "name": "ibm-cp-aiops",
"reconcileID": "281a1efa-dc33-4e86-ac4e-89a5d94d7725", "error": "admission webhook \"vautomationbase.kb.io\"
denied the request: Found 1 error(s):\n1. CartridgeRequirement exists. Cannot delete AutomationBase
instance\n"}
```

Solution:

1. Run the following command to find the name of the **ibm-common-service-operator** subscription.

   ```
   oc get subscription -A |grep ibm-common-service-operator-v3.23-ibm-operator-catalog-openshift-
   marketplace
   ```

   Example output:

   ```
   oc get subscription -A |grep ibm-common-service-operator-v3.23-ibm-operator-catalog-openshift-
   marketplace
   cp4aiops   ibm-common-service-operator-v3.23-ibm-operator-catalog-openshift-marketplace   ibm-common-
   service-operator   ibm-operator-catalog   v3.23
   ```

2. Run the following command to edit the **ibm-common-service-operator** subscription.

   ```
   oc edit subscription <ibm-common-service-operator-name> -n <project>
   ```

   Where

   - **<ibm-common-service-operator-name>** is the **ibm-common-service-operator** subscription returned by the previous step.
   - **<project>** is the project (namespace) where IBM Cloud Pak for AIOps is deployed.

   Example command:

   ```
   oc edit subscription  ibm-common-service-operator-v3.23-ibm-operator-catalog-openshift-marketplace -n
   cp4aiops
   ```

3. Remove the following line, and save your changes.

   ```
   startingCSV: ibm-common-service-operator.v3.23.3
   ```

After upgrade, policies do not process alerts due to problem with the aiops-ir-lifecycle-create-policies-job

This issue can occur if there is a problem connecting to Cassandra during upgrade. The **aiops-ir-lifecycle-create-policies-job** reports successful completion even it was unable to connect to Cassandra and run some of its policy-related queries. After upgrade policies do not process alerts, and error messages similar to the following are present in the **aiops-ir-lifecycle-create-policies-job's** log:

```
"name":"installdefaultpolicies","hostname":"aiops-ir-lifecycle-create-policies-job-
c8k7t","pid":42,"level":40,"err":{"message":"All host(s) tried for query failed.
```

```
"name":"installdefaultpolicies","hostname":"aiops-ir-lifecycle-create-policies-job-
c8k7t","pid":42,"level":50,"msg":"Failed to update policy with content
```

Solution: Restart the **aiops-ir-lifecycle-create-policies-job** by running the following command:

```
oc delete job aiops-ir-lifecycle-create-policies-job
```

Upgrade to 4.2.1 hangs, with IssueResolutionCore stuck in the "Update in progress" state.

When you upgrade to 4.2.1, an issue with the Redis operator can cause the upgrade to hang with IssueResolutionCore stuck in the "Update in progress" state. This issue does resolves without intervention. If you encounter this issue, wait for the issue to resolve, which can take up to a few

hours.

Metric anomaly alerts are no longer generated after upgrade

This issue can occur when connectivity to **kafka** is interrupted during upgrade.

Solution: Delete the **aiops-ir-analytics-metric-spark** pod to cause it to restart.

After upgrade, closed alerts are not cleared from the alert list

The job that is responsible for clearing closed alerts, **aiops-ir-core-ncodl-jobmgr**, did not restart properly after the upgrade.

Solution: Delete the **aiops-ir-core-ncodl-jobmgr** pod to cause it to restart.

Vault operator throws constraints not satisfiable error, and upgrade stalls

The upgrade procedure requires the AIOpsEdge deployment to be scaled down before IBM Cloud Pak for AIOps is upgraded. If this procedure was not followed, Red Hat OpenShift Operator Lifecycle Manager (OLM) reports the following error for Vault, an internal component which is removed in IBM Cloud Pak for AIOps 4.2.0.

```
- message: 'constraints not satisfiable: no operators found in channel v4.2 of package
     ibm-vault-operator in the catalog referenced by subscription vault, subscription
     vault exists'
   reason: ConstraintsNotSatisfiable
   status: "True"
   type: ResolutionFailed
```

Solution: Run the following command to manually update the channel in the **aiopsedge-base** OperandRegistry for vault to *v4.1*, to enable the upgrade to proceed.

```
oc patch OperandRegistry aiopsedge-base --type='json' -p='[{"op": "replace", "path":
"/spec/operators/1/channel", "value": "v4.1"}]' -n <project>
```

Where **<project>** is the project (namespace) where IBM Cloud Pak for AIOps is deployed.

New alerts from metric anomaly detection are not showing

Alerts from metric anomaly detection do not show if you did not complete all of the post upgrade steps for a previous upgrade of your deployment. You can rectify this by using the solution in After upgrade, Netcool and Metrics connections have a red status in the Connections UI.

After upgrade, Netcool and Metrics connections have a red status in the Connections UI

The Metrics and Netcool connections in the **Data & Tools Connections UI** have a red status after upgrading from 4.1.x to 4.2.x. This causes the connection pods to not upgrade to the latest version. You must use the following steps after upgrade so that the connection pods can upgrade to the latest version.

The following example shows an event that indicates the failure of the deployment update:

```
$ oc describe gitapp c18fce82-2a7b-45f2-8d36-d34417576fe7

<...output omitted...>
Events:
  Type     Reason           Age                     From    Message
  ----     ------           ----                    ----    -------
  Warning  ReconcileError   145m (x15005 over 29h)  GitApp  Deployables: failed to apply connector deployment
yaml to cluster: StatefulSet.apps "ibm-dyna-conn-9c02b6f2-6e37-4646-bee3-bafef95d9c2d" is invalid: spec:
Forbidden: updates to statefulset spec for fields other than 'replicas', 'template', 'updateStrategy',
'persistentVolumeClaimRetentionPolicy' and 'minReadySeconds' are forbidden
```

Solution: Download and run the **recreate-connector-storage.sh** script from github.com/IBM.

Log in to your Red Hat OpenShift Container Platform cluster, and then run the script on all connections, or on just one connection.

Running the script on all connections

1. (Optional) Check for any GitApp resources that may be stuck in Retrying phase.

   ```
   oc get gitapp
   ```

2. Run the recreate-connector-storage.sh script with --all argument to fix all defective connections.

   ```
   recreate-connector-storage.sh --namespace <project> --all
   ```

   Where **<project>** is the project (namespace) where IBM Cloud Pak for AIOps is deployed.

3. Verify that all GitApp resources statuses are now *Configured*

```
oc get gitapp
```

Running the script on a single connection

1. Run the following command to get the name of the connection.

   ```
   oc get connectorconfiguration --namespace <project>
   ```

   Where **<project>** is the project (namespace) where IBM Cloud Pak for AIOps is deployed.

   Example ouput:

   ```
   oc get connectorconfiguration appdhis --namespace cp4aiops
   ```

   ```
   NAME      DISPLAY NAME   TYPE          STATUS
   appdhis                  appdynamics   Running ConnectorConfiguration id: 7abfde13-6c2d-4c07-acd9-
   c50c346806df
   ```

2. Run the script for only the defective connection.

   ```
   recreate-connector-storage.sh --namespace <project> --connection <connection_config_name>
   ```

   Where:

   - **<project>** is the project (namespace) where IBM Cloud Pak for AIOps is deployed.
   - **<connection_config_name>** is the connection configuration name that was returned in the previous step.

   For example:

   ```
   ./recreate-connector-storage.sh --namespace cp4aiops --connection appdhis
   ```

3. Verify that the resource status for your connection is now *Configured*.

   ```
   oc get gitapp
   ```

Anomalies from log connections are not being detected

Anomalies are not detected after you upgrade to IBM Cloud Pak for AIOps version 4.2.1.

Solution: Resave the data connection.

1. From the IBM Cloud Pak Automation console console, expand the navigation menu and click **Define** > **Data and tool connections**.
2. Open the previously created connection and save it again.

Offline install or upgrade stuck because the 'oc ibm-pak generate mirror-manifests' command fails with 'no space left on device'

The **oc ibm-pak generate mirror-manifests $CASE_NAME $TARGET_REGISTRY --version $CASE_VERSION** command fails with a message similar to the following in **$IBMPAK_HOME/.ibm-pak/logs/oc-ibm_pak.log**:

```
write /tmp/render-unpack-4002583241/var/lib/rpm/Packages: no space left on device
```

Solution: The default temporary directory does not have enough space to run the **ibm-pak** tool. You must set the **TMPDIR** environment variable to a different directory with more space before running the **oc ibm-pak generate mirror-manifests** command.

```
TMPDIR=<new_temp_dir> oc ibm-pak generate mirror-manifests ${CASE_NAME} ${TARGET_REGISTRY} --version
${CASE_VERSION}
```

Where **<new_temp_dir>** is the path for a directory with more space.

After upgrading Red Hat OpenShift, AIOpsEdge is in an unhealthy state with a certification verification error

The secret **cp4waiops-connectors-deploy-cert-secret** has become invalid and is causing a certification verification error. An error similar to the following is seen in the log for the **aiopsedge-operator-controller-manager**:

```
2023-11-27T18:08:03.982Z ERROR controller.gitapp Reconciler error {"reconciler group":
"connectors.aiops.ibm.com", "reconciler kind": "GitApp", "name": "f1a461f4-bbd7-46d5-9439-8e71251449f3",
"namespace": "cp4waiops", "error": "Get "https://cp4waiops-connectors-deploy.cp4waiops.svc/git/cp4waiops-
connectors/info/refs?service=git-upload-pack\": tls: failed to verify certificate: x509: certificate signed
by unknown authority (possibly because of "crypto/rsa: verification error" while trying to verify candidate
authority certificate "openshift-service-serving-signer@1116814896")"}
```

Solution:

1. Delete the secret **cp4waiops-connectors-deploy-cert-secret**. It will be recreated automatically by the **AiopsEdge** operator.

   ```
   export PROJECT_CP4AIOPS=<project>
   oc delete secret cp4waiops-connectors-deploy-cert-secret -n ${PROJECT_CP4AIOPS}
   ```

   Where **<project>** is the project (namespace) that your IBM Cloud Pak for AIOps installation is deployed in.

2. Run the following commands to restart the **cp4waiops-connectors-deploy** and **aiopsedge-operator-controller-manager** pods.

   ```
   oc delete pod -n ${PROJECT_CP4AIOPS} $(oc get pod -n ${PROJECT_CP4AIOPS} | grep cp4waiops-connectors-deploy |awk '{print $1}')
   oc delete pod -n ${PROJECT_CP4AIOPS} $(oc get pod -n ${PROJECT_CP4AIOPS} | grep aiopsedge-operator-controller-manager |awk '{print $1}')
   ```

Troubleshooting clusters and nodes

Learn how to isolate and resolve problems that involve cluster and node failures.

- Merge Service timeout
- Temporal grouping policies not created after training historic events
- Illegal reflective access operation in Jobmanager pod log
- Kafka pods fail to restart
- Automation console not accessible after shutdown of backup cluster
- Log anomaly detection Kafka intermittent Issue, as Log anomaly detector abruptly stops making predictions
- Some connections reported in error and/or connection dataflow disable toggle fails to save with internal server error
- EDB Postgres cluster failure
- An EDB Postgres replica is in CrashLoopBackOff
- CouchDB pod reports a permission error when draining a node
- Embedded EDB PostgreSQL database license key expires on 1 October 2024

Merge Service timeout

The Merge Service polls Kafka for messages with a timeout of one second. This means that if there are no messages in the topic, the Merge Service waits one second before it checks again. However, this timeout value can cause issues if the Merge Service is not able to process the message within that time frame.

The Merge Service has a poll timeout of one second on consumers, whereas a higher value such as ten seconds or 100 seconds is needed.

Solution: You can increase the timeout period for the Merge Service by updating the **merge.kafkaConsumer.pollTimeout** parameter. The parameter specifies the amount of time (in milliseconds) that the Merge Service waits for a message before it checks again.

You can run the following **oc patch** command to set the **pollTimeout** value.

```
oc patch asmformations.asm.ibm.com --namespace $NAMESPACE --type='merge' --patch='{"spec":{"helmValuesASM": {"merge.kafkaConsumer.pollTimeout": "10000"}}}'
```

Example output:

```
asmformations.asm.ibm.com/asm patched
```

Verify that the value is updated by running the oc get deployment command.

```
oc get deployment noi-topology-merge -o yaml | grep POLL_TIMEOUT -A1
```

The following output is displayed.

```
      - name: KAFKA_CONSUMER_POLL_TIMEOUT
        value: "10000"
```

Temporal grouping policies not created after training historic events

Groups of related alerts that occur together (temporal groups) are not correlated automatically by the system. This issue can occur when temporal grouping policies are not created automatically due to communication issues between the aiops-ir-core-archiving pod and some Kafka topics.

Solution: Complete the following steps:

1. Run the following command, while logged in as an admin user in the project (namespace) where IBM Cloud Pak® for AIOps is running:

   ```
   kubectl logs $(kubectl get pods|grep aiops-ir-core-archiving|grep -v setup|awk '{print $1}') | grep "lib-rdkafka status"| tail -1
   ```

Example command output:

```
{
"name": "client.kafka",
"hostname": "aiops-ir-core-archiving-5dfd8fdcc5-l52g8",
"pid": 20,
"level": 30,
"brokerStates": {
   "0": "UP",
   "-1": "UP"
},
"partitionStates": {
   "cp4waiops-cartridge.irdatalayer.replay.alerts.0": "UP",
   "cp4waiops-cartridge.irdatalayer.replay.alerts.1": "UP",
   "cp4waiops-cartridge.irdatalayer.replay.alerts.2": "UP",
   "cp4waiops-cartridge.irdatalayer.replay.alerts.3": "UP",
   "cp4waiops-cartridge.irdatalayer.replay.alerts.4": "UP",
   "cp4waiops-cartridge.irdatalayer.replay.alerts.5": "UP"
},
"msg": "lib-rdkafka status",
"time": "2021-10-19T11:00:49.852Z",
"v": 0
}
```

2. If any of the broker or partition states are not up, restart the aiops-ir-core-archiving pod with the following command:

```
kubectl delete pod $(kubectl get pods|grep aiops-ir-core-archiving|grep -v setup|awk '{print $1}')
```

Illegal reflective access operation in Jobmanager pod log

Reports of illegal reflective access operation.

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by
org.apache.flink.shaded.akka.org.jboss.netty.util.internal.ByteBufferUtil (file:/opt/flink/lib/flink-
dist_2.11-1.13.2.jar) to method java.nio.DirectByteBuffer.cleaner()
WARNING: Please consider reporting this to the maintainers of
org.apache.flink.shaded.akka.org.jboss.netty.util.internal.ByteBufferUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
```

This warning is reported when the Flink library is run by Java 11. For more information, see https://issues.apache.org/jira/browse/FLINK-17524

Solution: No action required. This warning message can be ignored and does not impact functionality.

Kafka pods fail to restart

Kafka pods fail to start when a worker node goes down, as in the following example:

```
evtmanager-kafka-0                                                1/2     Running            55
15h
evtmanager-kafka-1                                                2/2     Running            0
16h
evtmanager-kafka-2                                                1/2     Running            0
15h
evtmanager-kafka-3                                                1/2     Running            0
15h
evtmanager-kafka-4                                                1/2     Running            0
15h
evtmanager-kafka-5                                                2/2     Running            0
15h
evtmanager-zookeeper-0                                            1/1     Running            0
16h
evtmanager-zookeeper-1                                            1/1     Running            0
15h
evtmanager-zookeeper-2                                            1/1     Running            0
15h
```

If the worker node where the Kafka pods are located goes down, the Kafka pods might fail to restart on another node.

Solution: Restart all zookeeper and Kafka pods:

```
oc delete pod {evtmanager-zookeeper-0} {evtmanager-zookeeper-1} {evtmanager-zookeeper-n} {evtmanager-kafka-0}
{evtmanager-kafka-1} {evtmanager-kafka-n}
```

Automation console not accessible after shutdown of backup cluster

Unable to access the IBM Cloud Pak Automation console on the backup cluster after a systematic shut down and start up of the backup cluster. The following error message is displayed:

```
"Document was encrypted with unknown key '<namespace>-<release name>-<timestamp>'"
```

Solution: Find and then delete the required databases through the API.

1. First, run the following command to retrieve the hostname:

   ```
   oc get route couchdb-georedundancy
   ```

   For this scenario, use the hostname `couchdb-georedundancy-geonoi.apps.geo01primary.myibm.com`.

2. Run the two `SECRET_NAME` commands to retrieve the `username` and `password` that are required to retrieve the database names.

   For Username:

   ```
   SECRET_NAME=$(oc get secret | grep couchdb-secret | awk '{ print $1 }'); kubectl get secret
   ${SECRET_NAME} -o json | grep "username" | cut -d : -f2 | cut -d '"' -f2 | base64 -d
   username
   ```

   For Password:

   ```
   SECRET_NAME=$(oc get secret | grep couchdb-secret | awk '{ print $1 }'); kubectl get secret
   ${SECRET_NAME} -o json | grep "password" | cut -d : -f2 | cut -d '"' -f2 | base64 -d
   password
   ```

3. Use the `hostname`, `username`, and `password` obtained in the previous steps to run the command to retrieve the database names:

   ```
   curl -sk -u "username:password" https://couchdb-georedundancy-
   geonoi.apps.geo01primary.myibm.com/_all_dbs | jq "."
   ```

   Example output:

   ```
   [
   "_global_changes",
   "_replicator",
   "_users",
   "collabopsuser",
   "emailrecipients",
   "genericproperties",
   "icp-63666439356237652d336263372d343030362d613461382d613733613739633731323535-rba-as",
   "icp-63666439356237652d336263372d343030362d613461382d613733613739633731323535-rba-rbs",
   "integration",
   "noi-drdb",
   "noi-osregdb",
   "normalizercfd95b7e-3bc7-4006-a4a8-a73a79c71255",
   "osb-map",
   "otc_omaas_broker",
   "rba-pdoc",
   "schedule",
   "tenant",
   "trainingjob"
   ]
   ```

   The relevant database name here is **collabopsuser**.

4. Run the following command to delete the database:

   ```
   curl -sk -u "root:netcool" -X DELETE "https://couchdb-georedundancy-
   geonoi.apps.geo01primary.myibm.com/${DATABASE_NAME}"
   ```

   Where `DATABASE_NAME` = `collabopsuser`

5. Restart the `cem-users` pod to recreate the database:

   ```
   oc get pods |grep cem-users
   <cem_users_pod_name>

   oc delete pod <cem_users_pod_name>
   ```

6. You can confirm the database is created by running the curl command from step 3:

   ```
   curl -sk -u "username:password" https://couchdb-georedundancy-
   geonoi.apps.geo01primary.myibm.com/_all_dbs | jq "."
   ```

LAD Kafka intermittent Issue, as Log anomaly detector abruptly stops making predictions

The log anomaly detector abruptly stops making predictions.

On checking the anomaly pod logs, you see error messages related to Kafka, and due to these intermittent issues, log anomaly stops consuming data from the windowed logs topic, even when there are continuous windows generated from the data preprocessing component.

An example of such an error in the logs is as follows:

```
[2022-03-19 14:28:34,233] [logprophet.logs.watson_aiops] [ERROR] An error has occurred while processing
message in batch mode. Error: 'NoneType' object is not iterable
[2022-03-19 14:28:34,234] [app.anomaly_detector.service_controller] [ERROR] Consumer failed with unhandled
exception : 'NoneType' object is not iterable
```

Solution: There is a two-step process to resume the flow:

1. Clear the windowed logs topic content. This step is required to clear any old data that is not yet consumed by the log anomaly detector. If you do not clear this data, when LAD is ready to make predictions, it will try to generate events for old data showing a lag. So, you need to disable the data flow, and clear data from windowed logs.
    - Disable data flow from data connection
    - Clear the windowed logs topic using following command

      ```
      oc edit kt cp4waiops-cartridge-windowed-logs-1000-1000
      ```

    - Set `retention.ms` to `1`.
    - Wait for a few minutes, and once the data is cleared and no more windows are generated, reset the 'retention.ms' to the original value
    - Enable the data flow in the data connection
2. Once the data is cleared using the above steps, restart the anomaly pod logs with the following command:

   ```
   oc get po | grep anomaly
   oc delete pod <anomaly-pod>
   ```

Some connections reported in error and/or connection dataflow disable toggle fails to save with internal server error

The symptoms sometime differ, but the most common symptom is that several connections are reported in error in the UI. Also it can be that the connection dataflow disable toggle fails to save with internal server error, which is due to a Flink problem in the HA topology.

Several connections are reported in error in the UI, or the connection dataflow disable toggle fails to save with internal server error, which is due to a Flink problem in HA topology. The following command shows the logs:

```
oc logs cp4waiops-eventprocessor-eve-29ee-ep-jobmanager-0 jobmanager
```

A sample of the root cause as taken from the cp4waiops-eventprocessor-eve-29ee-ep-jobmanager pods log is shown below:

```
2022-02-25 08:57:24,288 ERROR io.fabric8.kubernetes.client.extended.leaderelection.LeaderElector [] -
Exception occurred while acquiring lock 'ConfigMapLock: cp4aiops - cp4waiops-eventprocessor-eve-29ee-ep-
resourcemanager-leader (a1de269d-3fb9-49e0-96ab-0b4248ba046a)'
io.fabric8.kubernetes.client.KubernetesClientException: Operation: [get]  for kind: [ConfigMap]  with name:
[cp4waiops-eventprocessor-eve-29ee-ep-resourcemanager-leader]  in namespace: [cp4waiops]  failed.
```

Solution: The workaround is to restart all eventprocessor-eve pods.

```
oc project <project>
for n in $(oc get cm |grep job |awk '{print $1}')
do
  oc delete cm $n
done
oc delete pod cp4waiops-eventprocessor-eve-29ee-ep-jobmanager-0 cp4waiops-eventprocessor-eve-29ee-ep-
jobmanager-1
```

Where `<project>` is the project (namespace) where IBM Cloud Pak for AIOps is deployed.

EDB Postgres cluster failure

Run the following command on your cluster:

```
oc get clusters.postgresql.k8s.enterprisedb.io -n ibm-cp-aiops-edb-postgres -n <namespace>
```

Where `<namespace>` is the namespace that IBM Cloud Pak for AIOps is deployed in.

If the cluster *STATUS* does not show **`Cluster in healthy state`** and the *READY* count is 0, then the EDB Postgres cluster has failed.

EDB Postgres is a component that is used by IBM Cloud Pak for AIOps. A defect in EDB Postgres 1.18.6 and lower can cause the EDB Postgres cluster to fail. If the pod that hosts the primary instance of EDB Postgres is deleted and there is not a healthy EDB Postgres replica instance available to promote, then the EDB Postgres operator may not reschedule the pod hosting the primary instance.

Solution: If you encounter this problem, then you must contact IBM Support who can help you to determine and bring up the primary EDB Postgres node.

An EDB Postgres replica is in CrashLoopBackOff

One of the EDB Postgres instances in the EDB Postgres cluster has a state of **`CrashLoopBackOff`**, as in the following example where the second instance has failed.

```
oc get pod |grep ibm-cp-aiops-edb-postgres
```

```
ibm-cp-aiops-edb-postgres-1                                       1/1     Running            1                  5d15h
ibm-cp-aiops-edb-postgres-2                                       0/1     CrashLoopBackOff   198 (3m34s ago)    16h
ibm-cp-aiops-edb-postgres-3                                       1/1     Running            1                  5d15h
```

Entries similar to the following example can be seen in the Postgres pods' logs:

```
{"level":"info","ts":1677595436.8679705,"logger":"postgres","msg":"record","logging_pod":"ibm-cp-aiops-edb-
postgres-1","record":{"log_time":"2023-02-28 14:43:56.867
UTC","user_name":"streaming_replica","process_id":"90988","connection_from":"10.254.24.22:32852","session_id"
:"63fe132c.1636c","session_line_num":"1","command_tag":"idle","session_start_time":"2023-02-28 14:43:56
UTC","virtual_transaction_id":"13/0","transaction_id":"0","error_severity":"ERROR","sql_state_code":"58P01","
message":"requested WAL segment 000000030000000200000A0 has already been removed","query":"START_REPLICATION
2/A0000000 TIMELINE 2","application_name":"ibm-cp-aiops-edb-postgres-2","backend_type":"walsender"}}
```

A Postgres instance falls behind or is disconnected from the primary Postgres instance, and the primary instance removes or recycles the data that it requires. When the Postgres instance tries to restart, it fails because the data it requires is no longer available.

Solution: The failed Postgres replica is out of sync with the other replicas, and cannot recover on its own. Use the following commands to remove the failing Postgres instance and its associated PVCs, and cause a new replica to be created.

1. Install version 1.18.1 or higher of the **`cnp`** plugin. This is the EDB Postgres for Kubernetes plug-in. For more information about available EDB Postgres releases, see the EnterpriseDB documentation.

   Run the following command:

   ```
   curl -sSfL https://github.com/EnterpriseDB/kubectl-cnp/raw/main/install.sh | sudo sh -s -- -b
   /usr/local/bin v<version>
   ```

   Where **`<version>`** is the version of the plugin that you want to install, and is version 1.18.1 or higher.

   For example:

   ```
   curl -sSfL https://github.com/EnterpriseDB/kubectl-cnp/raw/main/install.sh | sudo sh -s -- -b
   /usr/local/bin v1.18.1
   ```

2. Find which of the Postgres replicas in your cluster is the primary.

   Run the following command:

   ```
   oc get clusters.postgresql.k8s.enterprisedb.io ibm-cp-aiops-edb-postgres
   ```

   Example output:

   ```
   oc get clusters.postgresql.k8s.enterprisedb.io
   NAME                                AGE     INSTANCES   READY   STATUS              PRIMARY
   ibm-cp-aiops-edb-postgres   7d17h   3           2       Upgrading cluster   ibm-cp-aiops-edb-postgres-1
   ```

   In the example output, **`ibm-cp-aiops-edb-postgres-1`** is listed under *PRIMARY*, and is the primary Postgres instance.

3. Find the number of the failed Postgres replica, which is in **`CrashLoopBackOff`**.

   Run the following command:

   ```
   oc get pod |grep ibm-cp-aiops-edb-postgres
   ```

   Example output:

   ```
   oc get pod |grep ibm-cp-aiops-edb-postgres
   ibm-cp-aiops-edb-postgres-1                                   1/1     Running            1
   5d15h
   ibm-cp-aiops-edb-postgres-2                                   0/1     CrashLoopBackOff   198 (3m34s ago)
   16h
   ```

```
ibm-cp-aiops-edb-postgres-3                                  1/1     Running            1
5d15h
```

In the example output, **ibm-cp-aiops-edb-postgres-2** is in **CrashLoopBackOff**, so the failed replica number is **2**.

4. Remove the failed Postgres pod and its associated PVC.

   Warning: Do not run the **oc cnp destroy** command against the primary Postgres replica, which you identified in step 2.

   Run the following command:

   ```
   oc cnp destroy ibm-cp-watson-aiops-edb-postgres <instance_id>
   ```

   Where **<instance_id>** is the failed Postgres replica number, which you identified in step 3.

   Example command:

   ```
   oc cnp destroy ibm-cp-watson-aiops-edb-postgres 2
   ```

   The **cnp** operator creates a new replica instance, and adds it to the EDB Postgres cluster.

5. After twenty minutes, verify that the Postgres cluster is fully functional again.

   1. Check that all of the Postgres pods are running.

      ```
      oc get pod |grep ibm-cp-aiops-edb-postgres
      ```

      Example output:

      ```
      ibm-cp-aiops-edb-postgres-1 1/1 Running 1 5d15h
      ibm-cp-aiops-edb-postgres-3 1/1 Running 1 5d15h
      ibm-cp-aiops-edb-postgres-4 1/1 Running 1 20m
      ```

   2. Run the following command, and verify that the output shows that all of the replicas are running and that the *STATUS* is **"Cluster in healthy state"**.

      ```
      oc get clusters.postgresql.k8s.enterprisedb.io ibm-cp-aiops-edb-postgres
      ```

      Example output:

      ```
      NAME                              AGE INSTANCES READY STATUS                     PRIMARY
      ibm-cp-aiops-edb-postgres 14d 3         3     Cluster in healthy state  ibm-cp-aiops-edb-postgres-1
      ```

CouchDB pod reports a permission error when draining a node

The **CouchDB** pod's log reports a permission error similar to the following example:

```
[error] 2022-12-02T01:24:20.726024Z couchdb@c-example-couchdbcluster-m-0.c-example-couchdbcluster-m
<0.13913.7> -------- Could not open file /data/db/shards/00000000-7fffffff/_users.1669801403.couch:
permission denied
```

Note: After a restore into a new namespace, similar symptoms can be seen. For more information about the solution for this scenario, see Restore fails with CouchDB pod in CrashLoopBackOff.

The way that Kubernetes mounts persistent volumes can cause this problem.

Solution: Delete the **CouchDB** pods that are not ready. When the replacement pods are created, the persistent volumes are mounted correctly.

1. Find the names of your **couchdb** pods.

   ```
   oc get pods -l app.kubernetes.io/name=IssueResolutionCoreCouchDB -n <namespace>
   ```

   Where **<namespace>** is the namespace that IBM Cloud Pak for AIOps is deployed in.

2. Delete all the **couchdb** pods that the previous step showed as not ready.

   ```
   oc delete pod <couchdb-pod> -n <namespace>
   ```

   Where:

   - **<namespace>** is the namespace that IBM Cloud Pak for AIOps is deployed in.
   - **<couchdb-pod>** is a couchdb pod that is not ready.

Embedded EDB PostgreSQL database license key expires on 1 October 2024

If your IBM Cloud Pak for AIOps deployment is older than v4.6.0, then you must update the embedded EDB PostgreSQL database license key before it expires on 1 October 2024. If the license expires, then the EDB PostgreSQL operator stops working and is unable to replace any replicas that go down, which then causes IBM Cloud Pak for AIOps downtime.

Symptoms of an expired license key: PostgreSQL cluster instances (pods) and data continue to be accessible when the EDB PostgreSQL database license key expires. However, the operator stops trying to reconcile the PostgreSQL cluster's status. As a result, the management of the cluster's status is suspended, including self-healing and high availability features like automated failover and primary instance switchover. An error similar to the following is seen in the custom resource:

```
status:
  licenseStatus:
    licenseStatus: Invalid license (license is expired)
    repositoryAccess: false
    valid: false
```

Solution: These steps can be used before the license key expires, or after expiration to recover your IBM Cloud Pak for AIOps deployment. You can update the EDB PostgreSQL database license key by upgrading to IBM Cloud Pak for AIOps 4.6.0 or higher, or by using the following manual process:

1. Find and export your project namespace.

   ```
   export PROJECT_CP4AIOPS=<project>
   ```

   Where **<project>** is the namespace (project) that IBM Cloud Pak for AIOps is deployed in.

2. Check the expiration date of the license key.

   ```
   oc get cluster.postgresql -n "${PROJECT_CP4AIOPS}" -o jsonpath='{range .items[*]}{.metadata.namespace}
   {"\t"}{.metadata.name}{"\t"}{.status.licenseStatus.licenseExpiration}{"\t"}
   {.status.licenseStatus.licenseStatus}{"\n"}{end}'
   ```

   Example output:

   ```
   aiops   common-service-db               Invalid license (license is expired)
   aiops   ibm-cp-aiops-edb-postgres       Invalid license (license is expired)
   aiops   zen-metastore-edb               Invalid license (license is expired)
   ```

   If **Invalid license** is shown then complete the remaining steps to update the license key.

3. Update the license key

   Online deployments:

   Run the following commands to update the license key image:

   ```
   oc get job create-postgres-license-config -n ibm-common-services -o yaml | \
   sed -e 's/operator.ibm.com\/opreq-control: "true"/operator.ibm.com\/opreq-control: "false"/' \
       -e 's|\(image: \).*|\1"cp.icr.io/cp/cpd/edb-postgres-license-
   provider@sha256:c1670e7dd93c1e65a6659ece644e44aa5c2150809ac1089e2fd6be37dceae4ce"|' \
       -e '/controller-uid:/d' | \
   oc replace --force -f - && \
   oc wait --for=condition=complete job/create-postgres-license-config -n ibm-common-services
   ```

   Offline deployments:

   1. Copy the new license key image to your private registry.

      ```
      export ENTITLED_REGISTRY_PASSWORD=<your IBM entitlement key>
      export TARGET_REGISTRY_HOST=<IP_or_FQDN_of_target_registry>
      export TARGET_REGISTRY_PORT=<port_number_of_target_registry>
      export TARGET_REGISTRY=$TARGET_REGISTRY_HOST:$TARGET_REGISTRY_PORT
      export TARGET_REGISTRY_USER=<username>
      export TARGET_REGISTRY_PASSWORD=<password>

      podman login cp.icr.io -u cp -p ${ENTITLED_REGISTRY_PASSWORD}
      podman login ${TARGET_REGISTRY} -u ${TARGET_REGISTRY_USER} -p ${TARGET_REGISTRY_PASSWORD}

      podman pull cp.icr.io/cp/cpd/edb-postgres-license-
      provider@sha256:c1670e7dd93c1e65a6659ece644e44aa5c2150809ac1089e2fd6be37dceae4ce
      podman tag cp.icr.io/cp/cpd/edb-postgres-license-provider cp.icr.io/cp/cpd/edb-postgres-license-
      provider:1600
      podman push cp.icr.io/cp/cpd/edb-postgres-license-provider:1600 ${TARGET_REGISTRY}/cp/cpd/edb-
      postgres-license-provider:1600
      ```

   2. Update the license key image.

```
oc get job create-postgres-license-config -n ibm-common-services -o yaml | \
sed -e 's/operator.ibm.com\/opreq-control: "true"/operator.ibm.com\/opreq-control: "false"/' \
    -e 's|\(image: \).*|\1"cp.icr.io/cp/cpd/edb-postgres-license-
provider@sha256:c1670e7dd93c1e65a6659ece644e44aa5c2150809ac1089e2fd6be37dceae4ce"|' \
    -e '/controller-uid:/d' | \
oc replace --force -f - && \
oc wait --for=condition=complete job/create-postgres-license-config -n ibm-common-services
```

4. Verify that the license key expiration date is updated.

   Run the following command to verify that the expiration date is **2999-12-31**.

   ```
   oc get cluster.postgresql -n "${PROJECT_CP4AIOPS}" -o jsonpath='{range .items[*]}{.metadata.namespace}
   {"\t"}{.metadata.name}{"\t"}{.status.licenseStatus.licenseExpiration}{"\t"}
   {.status.licenseStatus.licenseStatus}{"\n"}{end}'
   ```

   Example output for a deployment with a successfully modified expiration date:

   ```
   NAMESPACE   NAME                        LICENSE_EXPIRATION     LICENSE_STATUS
   cp4aiops    common-service-db           2999-12-31T00:00:00Z   Valid license (IBM - Data & Analytics
   (ibmda_cloudpak))
   cp4aiops    ibm-cp-aiops-edb-postgres   2999-12-31T00:00:00Z   Valid license (IBM - Data & Analytics
   (ibmda_cloudpak))
   cp4aiops    zen-metastore-edb           2999-12-31T00:00:00Z   Valid license (IBM - Data & Analytics
   (ibmda_cloudpak))
   ```

Troubleshooting connections

Review frequently encountered issues with creating and managing outgoing and incoming connections.

- Data and tool connections page
- Metrics connections
- Instana
- Kafka
- Netcool
- ServiceNow
- Splunk
- IBM Tivoli Netcool/Impact

Data and tool connections page

On viewing the Data and tool connections page, you might encounter the following issues:

- Deleting a connection does not remove the PVC for the connection
- Error displays when you activate Kafka, Humio, Splunk, LogDNA, ELK, PagerDuty, ServiceNow connections
- Connections on the Data and tool connections page show a red 'Unknown' status
- Connections-ui pod in CrashLoopBackOff on the Data and tool connections page
- Data and tool connections page becomes unresponsive after some time or shows an error
- Incorrect connection status on the Data and tool connections page
- The connector keeps showing *Error* or *Retrying* and connector pod is stuck in *Pending*
- Ansible Automation Controller integration not running due to prohibited egress

Deleting a connection does not remove the PVC for the connection

For StatefulSet type connections, when the connection is deleted the Persistent Volume Claim (PVC) resource for the connection is not automatically deleted. You must manually delete the resource with the Red Hat OpenShift **oc** CLI or with the Red Hat OpenShift UI.

This behavior occurs because the StatefulSet PVC is created by specifying the **volumeClaimTemplate**, but the created PVC is not automatically deleted by Red Hat OpenShift when the StatefulSet resource is deleted.

To delete a PVC resource with the Red Hat OpenShift **oc** CLI, complete the following steps:

1. Get the list of PVC resources for your current connections:

   ```
   oc get pvc | grep 'conn'
   ```

2. Select the PVC resource that you want to delete and record the name.

3. Delete the PVC resource:

```
oc delete pvc <pvc-name>
```

Where **<pvc-name>** is the name of the PVC resource that you want to delete.

Error displays when you activate Kafka, Humio, Splunk, LogDNA, ELK, PagerDuty, ServiceNow connections

When you activate a Kafka, Falcon LogScale, Splunk, LogDNA, ELK, PagerDuty, ServiceNow connection in the console, an error can occur when the required Flink task slots to run the connection are not available. When this error occurs, the following message is displayed:

```
Failure: Not enough task slots to activate the connection.
```

This error occurs as there is a finite number of Flink task slots to use for creating connections to remote resources. The error message displays the number of slots that are needed to support the current active connection set, along with the total number of slots available.

Solution: To activate the connection, you must either reduce the number of slots that are needed by this (or another) connection, or allocate more slots. To decrease the number of slots needed by a connection, consider reducing the "base parallelism" or "logs per second" value for the connection. This reduced value decreases the rate at which data flows through the connection, making more slots available to other connections. To allocate more slots, see Increasing data streaming capacity.

Connections on the Data and tool connections page show a red 'Unknown' status

All connections are showing red 'Unknown' connection status and data is not flowing after creating a new connection, such as a LogDNA connection. For the new connection if you enable 'Data collection', you can see that the data is not flowing, and the connection shows the same red 'Unknown' status in the Data collection status field instead of the expected 'Running' status. The common cause for this is unstable Kafka in the background.

Solution: If this issue occurs, restart the bridge and operator pod.

Connections-ui pod in CrashLoopBackOff on the Data and tool connections page

You can encounter an intermittent issue that causes the Data and tool connections page to fail and become unavailable for viewing.

Solution: If the page does not automatically recover, you can increase the timeout setting for the **aimanager-aio-controller** deployment to help prevent the issue. To increase this setting, edit the **aimanager-aio-controller** deployment (**liveliness** and **healthcheck** probe) and set it to a longer timeout of say 15 seconds, up from the default 5 seconds. This change can stop the **Connections-ui** pod from going into CrashLoopBackOff.

1. Check the **aimanager-aio-controller**

   ```
   oc get pods | grep aio-controller
   aimanager-aio-controller-6d9f68d5f5-8dkk9                         1/1     Running     0
   27h
   ```

2. Edit the **aimanager-aio-controller**'s deployment

   ```
   oc edit deployment aimanager-aio-controller
   ```

3. Edit these 2 variables (**timeoutSeconds**) for both the **livenessProbe** and the **readinessProbe** to 15 seconds:

   ```
   livenessProbe
   - timeoutSeconds
   readinessProbe
   - timeoutSeconds
   ```

Data and tool connections page becomes unresponsive after some time or shows an error

If you are using IBM Cloud Pak for AIOps, you might encounter an unresponsive Data and tool connections page around 24 hours after you install your cluster. The connections page might take a long time to load and then display no connections despite configuring them earlier. Another symptom of this problem is that you might fail to create a connection and see the following error:

```
Failed submitting creating request
```

Solution: To ensure the Data and tool connections page works normally, delete the **aimanager-aio-controller** pod:

```
for n in $(oc get pods |grep aimanager-aio-controller |awk '{print $1}')
do
oc delete pod $n
done
```

Within one to two minutes of running the script, the pod will automatically restart and the Data and tool connections page will work as usual.

Incorrect connection status on the Data and tool connections page

You might encounter an incorrect connection status for all gRPC connectors (Instana, Netcool, AppDynamics, Dynatrace, AWS CloudWatch, Splunk, Zabbix, and New Relic) on the Data and tool connections page. The connection status might appear as *Unknown* when it must be *Running*. The incorrect status might be the result of the pod connector-bridge being in CrashLoopBackOff state and restarting multiple times due to an unstable Kafka connection.

Solution: If this issue occurs, check the Kafka and zookeeper pods for any restarts. Kafka connection becomes unstable and restarts when the PVC used for Kafka is full. Review the current capacity of the PVC and increase the capacity as needed to resolve the issue. For more information about adjusting the size of the PVC, see Increasing Kafka PVC.

The connector keeps showing *Error* or *Retrying* and connector pod is stuck in *Pending*

On the Data and Connections UI page, a connector keeps showing *Error* or *Retrying* and the connector pod is stuck in *Pending*. This issue occurs because the Persistent Volume Claim (PVC) is requesting a volume from a storage provider that does not support the **`ReadWriteOnce`** (RWO) access mode.

Solution: You can use either one of the following methods to unblock the stuck connector pod:

- Change the default storage class

Or

- Re-create the Persistent Volume Claim with a different storage class

Method 1: Change the default storage class. Review the default storage class in the cluster and change to a storage class that supports **`ReadWriteOnce`** (RWO) access mode, such as file storage types.

Note: This method should resolve connector PVC issues but since it changes the global default storage class type in the cluster, it might also affect other workloads that rely on the default storage class settings when requesting a volume.

1. Run the following command to determine which is the current default storage class.

   ```
   oc get storageclass
   ```

2. Note the name of the default storage class **`(default)`**.

3. Change the default storage class to one that supports the **`ReadWriteOnce`** (RWO) access mode like file storages. The following command sets the **`storageclass.kubernetes.io/is-default-class`** annotation to **`true`** on the new storage class and sets the annotation to **`false`** on the old storage class. (Change the **`<new-default-storageclass-name>`** and **`<old-default-storageclass-name>`** with the actual storage class names in your cluster.)

   ```
   oc patch storageclass <old-default-storageclass-name> -p '{"metadata": {"annotations":
   {"storageclass.kubernetes.io/is-default-class":"false"}}}'
   oc patch storageclass <new-default-storageclass-name> -p '{"metadata": {"annotations":
   {"storageclass.kubernetes.io/is-default-class":"true"}}}'
   ```

4. Verify that the default storage class is updated by running

   ```
   oc get storageclass
   ```

5. Log in to the Automation console and navigate to the **Data and Connections** page.

6. Delete the existing connection, then reinstall it.

7. Verify that the connection enters a *Running* state.

Method 2: Re-create the Persistent Volume Claim with a different storage class. Delete the existing PVC and re-create a new one with the same name that uses a storage class that supports the **`ReadWriteOnce`** (RWO) access mode.

Note: This method updates only a single connector instance. Repeat this step for all connectors unable to request a persistent volume, due to the incorrect storage class type.

1. Get the PVC name of the connector. You can list all PVCs by using **`oc get pvc`**. Look for the one that is still in *Pending* state with your connector name.

2. Delete the existing PVC and re-create a new one with the same name. Review the following sample command as reference. Update the **`<connector-pvc-name>`** with the PVC name from the previous step and set **`<storage-class-name>`** with the storage class that supports the required **`ReadWriteOnce`** (RWO) access mode.

   ```
   # Set the PVC name
   CONNECTOR_PVC_NAME=<connector-pvc-name>

   # Set new storage class
   STORAGE_CLASS=<new-storage-class>

   CONNECTOR_APP=$(oc get pvc $CONNECTOR_PVC_NAME -o jsonpath='{.metadata.labels.app}')
   CONNECTOR_ID=$(oc get pvc $CONNECTOR_PVC_NAME -o jsonpath='{.metadata.labels.instance}')
   PVC_SIZE=$(oc get pvc $CONNECTOR_PVC_NAME -o jsonpath='{.spec.resources.requests.storage}')

   oc delete pvc $CONNECTOR_PVC_NAME && \
   cat << EOF | tee >(oc apply -f -) | cat
   apiVersion: v1
   kind: PersistentVolumeClaim
   ```

```
metadata:
    name: $CONNECTOR_PVC_NAME
    labels:
        app: $CONNECTOR_APP
        app.kubernetes.io/instance: aiopsedge
        app.kubernetes.io/managed-by: aiopsedge-operator
        app.kubernetes.io/name: ibm-aiops-edge
        app.kubernetes.io/part-of: ibm-aiops-edge
        instance: $CONNECTOR_ID

spec:
    accessModes:
    - ReadWriteOnce
    resources:
        requests:
            storage: $PVC_SIZE
    storageClassName: $STORAGE_CLASS
    volumeMode: Filesystem
EOF
```

For example,

```
# Set the PVC name
CONNECTOR_PVC_NAME=netcool-vct-netcool-conn-325fba02-440f-4135-bd00-8560bcf5eda1-0

# Set new storage class
STORAGE_CLASS=rook-cephfs

CONNECTOR_APP=$(oc get pvc CONNECTOR_PVC_NAME -o jsonpath='{.metadata.labels.app}')
CONNECTOR_ID=$(oc get pvc CONNECTOR_PVC_NAME -o jsonpath='{.metadata.labels.instance}')

oc delete pvc $CONNECTOR_PVC_NAME && \
cat << EOF | tee >(oc apply -f -) | cat
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: $CONNECTOR_PVC_NAME
    labels:
        app: $CONNECTOR_APP
        app.kubernetes.io/instance: aiopsedge
        app.kubernetes.io/managed-by: aiopsedge-operator
        app.kubernetes.io/name: ibm-aiops-edge
        app.kubernetes.io/part-of: ibm-aiops-edge
        instance: $CONNECTOR_ID

spec:
    accessModes:
    - ReadWriteOnce
    resources:
        requests:
            storage: 50Mi
    storageClassName: $STORAGE_CLASS
    volumeMode: Filesystem
EOF
```

3. Verify that the connector pod transitions to **`Running`** state.
4. If necessary, repeat the steps above on any other connector that is producing the same problem.

Metrics connections

If you are creating, or have, a metrics connection, you might encounter the following issue:

Metrics connection is unable to pull further historical data when historical mode is completed

When a metrics connection (for example Dynatrace, Appdynamics, Zabbix, New Relic, Splunk, or AWS) has completed its historical data pulling, historical mode can never be restarted again even if the connection configuration is edited or updated from the UI to specify a different start date, end date or a different set of metrics.

Solution: If you need to pull historical data again for another set of metrics or different dates, you can create a new connection.

Instana connections

If you are creating, or have, an Instana connection, you might encounter the following issues:

- Connection is unable to update the metric list in the UI
- IBM Cloud Pak for AIOps cannot close some alerts when an Instana connection exists

- After a repair of Instana, alerts in the Instana application do not associate with the corresponding IBM Cloud Pak for AIOps application
- Due to an out-of-memory error the Instana connector pod keeps restarting when retrieving data from Instana

Connection is unable to update the metric list in the UI

The Instana Connection UI page displays a list of metrics from Instana that are updated when the Instana Connector pod runs. Some Instana hosts can have many technology plug-ins installed, which collect many built-in and custom metrics. These metrics can cause the Instana Connector to exceed the allowed message limit when the connection sends an update to the UI. If this limit is exceeded, the Instana Connector pod logs the following message:

```
0000004d Connector     I   UI configuration patch exceeds the safe message limit, skip sending configuration
patch. The metrics list in the UI could not be updated with the actual list of technologies and metrics.
Please configure more filter patterns to reduce the number of metrics to query.
```

The Connector logs each technology plug-in and the number of metrics that are included or filtered.

Solution: Apply additional filter patterns in the Instana configmap for the connector to exclude when querying the metric list from Instana.

1. Get the Instana pod logs to determine which technology plug-in contains many metrics

   ```
   oc logs $(oc get pod | grep 'instana-conn' | awk '{print $1}') | grep -e "Plugin \'.*\' \(.*\)"
   ```

   The following example log indicates that a **aceMessageFlow** technology plug-in that contains a large list of metrics, and that is a candidate to filter.

   Tip: Look for any technology plug-ins that contain more than 100 metrics.

   ```
   0000004d InstanaInstan I Plugin 'ACE Message Flow' (aceMessageFlow) metrics: 2,666 included, 0 filtered.
   ```

2. Determine the available metrics names by technology from the Instana API or from your Instana administrator. To use the Instana API, you can use a command similar to the following sample command, which queries **aceMessageFlow** metrics:

   ```
   INSTANA_ENDPOINT=<your Instana host>
   INSTANA_TOKEN=<your Instana API token>
   curl --request GET -s  --url ${INSTANA_ENDPOINT}/api/infrastructure-
   monitoring/catalog/metrics/aceMessageFlow \
   --header "authorization: apiToken ${INSTANA_TOKEN}" \
   | jq 'group_by(.pluginId)[]|  {(.[0].pluginId): [.[] | .metricId] | sort}'
   ```

3. Create a pattern to filter metrics from the output of the previous command. For example, **flowNodes\\..*** to filter all metrics with **flowNodes.** prefix.

4. Get the Instana Connector configmap name to add the filter pattern from the previous step.

   Note: If multiple Instana connections exist, select the configmap with the UID that matches the pod name.

   ```
   oc get configmap | grep instana-connector
   ```

5. Extract the **exclude-filter-patterns** from the configmap. This command saves the content of the configmap into the local file:

   ```
   oc extract configmap/ibm-grpc-instana-connector-d59eea9f-501d-4bd1-b3a3-6f98b49b0930 --keys=exclude-
   filter-patterns --to=.
   ```

6. Edit the local **exclude-filter-patterns** file to add the metric pattern in addition to the default ones that are defined in the **filters** array.

   Note: To escape a slash **** character, you must use double ****. For example:

   ```
   {"filters":["fs\\./dev/.*","fs_mount\\./.*","metrics\\.counters\\..*\\..*\\..*","(.*\\.)?
   metrics\\.gauges\\..*\\..*\\..*","metrics\\.meters\\..*","(.*\\.)?
   metrics\\.timers\\..*","prometheus\\.metrics.*","flowNodes\\..*"]}
   ```

7. Apply the change to the configmap:

   ```
   oc set data configmap/ibm-grpc-instana-connector-d59eea9f-501d-4bd1-b3a3-6f98b49b0930 --from-
   file=exclude-filter-patterns
   ```

8. Verify that the new filter pattern is loaded by the pod, and that the number of metrics that are filtered for the technology plug-in is changed.

   ```
   oc logs $(oc get pod | grep 'instana-conn' | awk '{print $1}') | grep 'Loaded filter patterns from
   ConfigMap'
   ```

9. Verify that the Instana Connector can send an update to the UI successfully:

   ```
   oc logs $(oc get pod | grep 'instana-conn' | awk '{print $1}') | grep -e 'Status message .* write with
   configuration patch .* successful.'
   ```

10. In the *Edit Instana Connection* UI page, the list of technologies and metrics list is now updated.

IBM Cloud Pak for AIOps cannot close some alerts when an Instana connection exists

When a connection with Instana exists, IBM Cloud Pak for AIOps might not be able to close some alerts. Instead, you might need to check the status for the associated event within the Instana UI and clear the alert manually.

Solution: To find the alert URL for clearing the alert, complete the following actions:

1. From the IBM Cloud Pak Automation console, open the *Incidents and alerts* tool.
2. Select the alert in the table to display the Alert details side panel for that alert. The Information section is open by default in the side panel.
3. Copy the URL from links and Paste the URL into your browser.
4. If you are an Instana authorized user, you see the associated event in the Instana UI.

If the event is not active and has "end-time" for the event in the Instana UI, clear the alert:

1. Click *Alert* to see the alert details.
2. Click *Clear Alert* to clear the alert.

After a repair of Instana, alerts in the Instana application do not associate with the corresponding IBM Cloud Pak for AIOps application

When you restart Instana post repair, alerts in the Instana application do not map with the corresponding application in IBM Cloud Pak for AIOps. The reason is that Instana creates new identifiers for the resources on its application on restarting, without updating the topology in IBM Cloud Pak for AIOps. As a result, the IDs of the resources do not match leading to the disassociation of alerts.

Solution: To map the alerts, you must complete the following steps:

1. Clean up the application in Instana.
2. Re-create the application after a few minutes.

To clean up and recreate applications in Instana, see Getting started with Instana.

Due to an out-of-memory error the Instana connector pod keeps restarting when retrieving data from Instana

When the Instana connector receives a large amount of data from one or more Instana APIs while the connector is observing many resources, and there is insufficient memory to process this large amount of data, an out of memory error can occur.

Solution: If this error occurs, you can apply custom memory settings to allocate more memory to the Instana connector pod by increasing the memory limit setting. This allocation can only be done post-installation as the connector must exist before applying the custom setting. To apply custom resource settings, follow the steps to patch the Instana Connector:

Prerequisite: Ensure you have the **yq** and **jq** packages installed on your system before running the commands below. To get installation instructions for these packages, refer to the documentation or website for **yq** and **jq**.

1. From a command line, change to your IBM Cloud Pak for AIOps project (namespace):

   ```
   namespace=<aiops-namespace>
   oc project $namespace
   ```

2. Get the display name for the Instana connection you would like to increase resources for from the command line:

   ```
   oc get connectorconfiguration
   ```

3. Set the required variables:

   ```
   connectionname=<instana-connection-name>
   connconfig=$(oc get connectorconfiguration --no-headers | grep $connectionname | awk '{print $1}')
   connconfiguid=$(oc get connectorconfiguration $connconfig -o jsonpath='{.metadata.uid}')
   gitappname=$(oc get gitapp -l connectors.aiops.ibm.com/connection-id==$connconfiguid --no-headers | awk
   '{print $1}')
   ```

   Note: Update the **connectionname** variable with your Instana display name from the command above.

4. Run this command to create a variable with a patch specification for the Instana connection. This will apply only to one Instana connection at a time. This command should not have an output. Copy this command directly from the documentation and do not paste it into a text editor because this can affect the format.

   ```
   jsonpatch=$(yq -o json <<EOF
       patch: |-
         apiVersion: apps/v1
         kind: Deployment
         metadata:
           name: ibm-grpc-instana-connector
         spec:
           template:
   ```

```
          spec:
            containers:
            - name: ibm-grpc-instana-connector
              resources:
                requests:
                  memory: 1536Mi
                  ephemeral-storage: 2Gi
                limits:
                  memory: 8Gi
                  ephemeral-storage: 4Gi
    target:
      group: apps
      kind: Deployment
      name: ibm-grpc-instana-connector
      version: v1
EOF
)
```

5. Run the following command to apply the custom resource setting that you created in the previous step:

   ```
   oc get gitapp $gitappname  -n $namespace -o json | jq ".spec.components[0].kustomization.patches += [$jsonpatch]" | oc apply -f -
   ```

   Note: If you see an error message when running this command that states there was a configuration conflict, try running the command a second time.

6. It takes a moment for the changes to show in the connector pod.

Note: If the memory limit is not updated for the Instana connector pod after a few minutes, run `oc get gitapp $gitappname` and check to see if there are any errors. If there is an error stating that the YAML is invalid, run the command below to revert the patch and go through all steps again. The cause of this error is that the format of the commands may have been altered by a text editor and should be copied directly from the documentation and run directly in the terminal.

Revert the custom resource settings to default resource settings:

To revert from custom to default resource settings, run the following command:

```
oc patch gitapp $gitappname -n $namespace --type='json' -p="[{'op': 'remove', 'path': '/spec/components/0/kustomization/patches'}]"
```

Kafka connections

If you are creating, or have, a Kafka connection, you might encounter the following issues:

- Kafka events not displayed in IBM Cloud Pak for AIOps
- After copying log data into a Kafka cluster, verify that the data is imported into Elasticsearch

Kafka events not displayed in IBM Cloud Pak for AIOps

You cannot see Kafka events in IBM Cloud Pak for AIOps after successfully creating a Kafka connection. This failure occurs when the Kafka key or message violates the format requirements. The `MismatchedInputException` error in the task manager pod log indicates this formatting error.

Solution: To ensure that the Kafka events display in IBM Cloud Pak for AIOps, check whether the following conditions are met:

1. The key in Kafka payload is null.

   Note: The key is determined by Kafka producer. Most producer tools use null object as key unless the producer is programmed to assign value.

2. The message in Kafka payload is a JSON message that satisfies the following requirements:

   - Exists in the format of the selected data source.
   - Contains sufficient data for the mapping.

After copying log data into a Kafka cluster, verify that the data is imported into Elasticsearch

To publish log data, you can copy it into a Kafka cluster. For more information, see Copying log data into the Kafka cluster. Then, you can check whether the log data is imported into Elasticsearch.

Solution: Verify that the log data is imported into Elasticsearch by getting shell access to the `ai-platform-api-server` pod. Run the following commands from the namespace where IBM Cloud Pak for AIOps is installed:

```
oc exec -it $(oc get pods | grep ai-platform-api-server | awk '{print $1}') -- /bin/bash
```

Example:

```
curl -X GET -u $ES_USERNAME:$ES_PASSWORD $ES_URL/_cat/indices -k | sort
```

Example output:

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4710  100  4710    0     0  28373      0 --:--:-- --:--:-- --:--:-- 28373
green  open .security                          L0Gou7AXSC6PyuZ8-skGRQ 1 0    1 0   4.6kb   4.6kb
green  open aiops-searchservice_v10           Yml s6nL3S-KP4vC4ROdRRw 5 0    0 0     1kb     1kb
yellow open 1000-1000-20200803-logtrain      QImZoOd7QiuM3mfxUIdFAA 8 2   24 0  34.5kb  34.5kb
yellow open 1000-1000-20200804-logtrain      54m5F8K6Q3iIbSZ4KvMlYQ 8 2  286 0  80.8kb  80.8kb
yellow open 1000-1000-20200805-logtrain      I_GhHFz9Th62KV_7wGwyMA 8 2  137 0  61.1kb  61.1kb
yellow open 1000-1000-20200806-logtrain      7HZcyCaLRa2VoOwJrH7Ftg 8 2  292 0  76.2kb  76.2kb
yellow open 1000-1000-20200807-logtrain      LzUFhGcGSs-Rp_uJtw17sw 8 2  686 0 126.1kb 126.1kb
yellow open 1000-1000-20200808-logtrain      HR7n9NIATnm9MOjgnq2K_Q 8 2  268 0    77kb    77kb
yellow open 1000-1000-20200809-logtrain      NSU1X31OTq2coxdo-_I1rQ 8 2   48 0  48.7kb  48.7kb
yellow open 1000-1000-20200810-logtrain      2M54aDMXRyOp_06XnInTEg 8 2  885 0 140.3kb 140.3kb
yellow open 1000-1000-20200811-logtrain      pQjyF1pQTX6GT70WY7T7ZQ 8 2  797 0 132.2kb 132.2kb
yellow open 1000-1000-20200812-logtrain      U0YasxsqTpuricpyyfQ62Q 8 2 1294 0 201.6kb 201.6kb
yellow open 1000-1000-20200813-logtrain      pZA79rSSTYeTrsRuT8K44g 8 2 1155 0 178.3kb 178.3kb
yellow open 1000-1000-20200814-logtrain      eu5qqE31TcGtz5yjVKxlnQ 8 2  720 0   134kb   134kb
yellow open 1000-1000-20200815-logtrain      rKHrLUooSP-lx6r_ettalQ 8 2   42 0  46.2kb  46.2kb
yellow open 1000-1000-20200816-logtrain      0bRKhqgdQviTBc0FnJ4pvg 8 2   58 0  49.9kb  49.9kb
yellow open 1000-1000-20200817-logtrain      PMrlHzjZQymb5hXiZanhOw 8 2  287 0  99.3kb  99.3kb
yellow open 1000-1000-20200818-logtrain      ItLXqZIxQWy7pirLCPcfKw 8 2  483 0 103.7kb 103.7kb
yellow open 1000-1000-20200819-logtrain      W2ti9RlVTRiW3ZpynEZ8wQ 8 2  879 0 141.8kb 141.8kb
yellow open 1000-1000-20200820-logtrain      Zsz7Vd0rQpSuVwbUi4AfQw 8 2  488 0  98.7kb  98.7kb
yellow open 1000-1000-20200821-logtrain      VVcsZ1HEQJi08WrjX6CJ_Q 8 2  268 0  43.4kb  43.4kb
yellow open algorithmregistry                wLjGMGL4SI-weK666yr8pA 1 1    8 0  26.2kb  26.2kb
yellow open cp4waiops-cartridge-poli.....    6dQclBWSQA6FR3yRurpxVQ 1 1   17 0  66.8kb  66.8kb
yellow open dataset                          TA3TB-MNTkScyxDNZxuEcw 1 1    0 0    208b    208b
yellow open postchecktrainingdetails         G8lHdh9WQ8OxIHhOAwBUnw 1 1    0 0    208b    208b
yellow open prechecktrainingdetails          WA00PRwuTgWLsQNvcBcoug 1 1    0 0    208b    208b
yellow open trainingdefinition               dC2UuOLiR_qGiax-8tEfEg 1 1    0 0    208b    208b
yellow open trainingrun                      fRqCLpQ1TCi_kt4zzNvwDA 1 1    0 0    208b    208b
yellow open trainingsrunning                 gzyQzqP7SJuy9dbUZI1dVA 1 1    0 0    208b    208b
yellow open trainingstatus                   AiLkAg0yRQKzzIpbsk0CCw 1 1    0 0    208b    208b
```

Netcool connections

If you are creating, or have, a Netcool connection, you might encounter the following issues:

- Runbook automation pod "rba-rbs" crashes during an alert burst, for example, while connecting to Netcool
- Setting values for mandatory fields for events sent to the embedded ObjectServer

Runbook automation pod "rba-rbs" crashes during an alert burst, for example, while connecting to Netcool

The "rba-rbs" pod crashes and gets restarted automatically, and most of the fully automated runbooks that are associated with the new alerts are not run. This restart behavior might be observed multiple times until all the events from the event burst have been ingested, and the system reaches its regular state of operation again.

Solution: During an event burst like the one that occurs when connecting to a Netcool server, all enabled policies are evaluated, and for the matching events, the associated fully automated runbooks are started. If many matching events exist, these events might result in a high load of concurrent runbook invocations that the runbook service rba-rbs is not able to handle. The system recovers automatically. However, some runbooks will not be run during the burst.

Setting values for mandatory fields for events sent to the embedded ObjectServer

If you are manually inserting events into the embedded ObjectServer using `nco_sql`, you must set values for the following fields so that the events can be processed correctly:

- `Node`
- `AlertKey`
- `AlertGroup`
- `Manager`

Note: This is only a concern if you are creating an event manually using an insert statement. All IBM Netcool Operations Insight probes will set all required columns.

ServiceNow integrations

Incidents aren't created

Incidents might not be created for various reasons.

Solution: Complete the following checks to help ensure that incidents can be created.

1. Validate the ServiceNow URL and credentials. Go to the ServiceNow integration page, select the integration, and select **Test connection**. If the test fails, check that the URL and credentials are correct.
2. Make sure that the versions of Cloud Pak for AIOps and the IBM ServiceNow marketplace applications are compatible. For more information about which version of Cloud Pak for AIOps is in use, see Prerequisites in Creating a ServiceNow integration.
3. Make sure that a policy is configured to use the existing ServiceNow connection. For more information about creating policies, see Promote alerts to an incident.
4. Make sure that the ServiceNow integration user has the correct permissions. For more information, see Managing roles.
5. Incident tables can have business rules that prevent incident creation. Check business rules and change them so that they don't affect incident tables.

Splunk connections

If you are creating, or have, a Splunk connection, you might encounter the following issues while creating or editing the connection:

- All fields in the Splunk form reset if edited while loading metrics
- Splunk connections that are created before upgrade fail to update after upgrade

All fields in the Splunk form reset if edited while loading metrics

If you edit the fields in the Splunk form while the metrics are loading in the connector console, then you cannot save the changes made to the form.

Solution: To avoid this issue, do not modify the fields in the Splunk form while metrics are loading. If you do so, then redo the changes after the metrics are loaded to save the edits.

Splunk connections that are created before upgrade fail to update after upgrade

On upgrading from 3.5.0, you might not be able to enable or disable log or metric data flows on Splunk connections that were created before upgrade. This issue can occur intermittently.

Solution: When this issue occurs, you can refresh the console and try again after some time. Alternatively, you can create a new connection and later delete the Splunk connection that was defined before upgrade.

IBM Tivoli Netcool/Impact connections

If you are creating, or have, a IBM Tivoli Netcool/Impact connection, you might encounter the following issues:

- IBM Tivoli Netcool/Impact Connection status shows "Retrying"
- IBM Tivoli Netcool/Impact connection stops event processing with exceptions
- IBM Tivoli Netcool/Impact stops sending data to widgets in a JazzSM dashboard

IBM Tivoli Netcool/Impact Connection status shows "Retrying"

After Creating IBM Tivoli Netcool/Impact connections, you might see a **Connection Orchestration** status of "Done", but the **IBM Tivoli Netcool/Impact Connection** shows an error with a status of "Retrying".

The reason for this error might be a locked IBM Tivoli Netcool/Impact file that is preventing the creation of the AIOps data model in IBM Tivoli Netcool/Impact. If this is the issue, the **impactserver-errors** logs in IBM Tivoli Netcool/Impact will display an error similar to the following:

```
30 Jun 2023 03:27:21,137 ERROR [DataModelUIResource] File etc/NCIP_datasourcelist is locked by user: root.
```

Solution: Locate and delete a file called **<servername>_versioncontrol.locks** in IBM Tivoli Netcool/Impact and restart the IBM Tivoli Netcool/Impact server and GUI.

1. Log in to the IBM Tivoli Netcool/Impact host and navigate to the **$IMPACT_HOME/etc** directory. Locate a file called **<servername>_versioncontrol.locks**. For example, **NCIP_versioncontrol.locks**.
2. Delete the file.
3. Restart the Impact server and GUI:

```
$IMPACT_HOME/bin/stopImpactServer.sh
$IMPACT_HOME/bin/stopGUIServer.sh

$IMPACT_HOME/bin/startImpactServer.sh
$IMPACT_HOME/bin/startGUIServer.sh
```

The status of the **IBM Tivoli Netcool/Impact Connection** should switch to **Running** in Cloud Pak for AIOps.

IBM Tivoli Netcool/Impact connection stops event processing with exceptions

If you have an IBM Tivoli Netcool/Impact connection, you can encounter an issue where the connection temporarily stops processing during the sending of an event stream to IBM Cloud Pak for AIOps.

This issue can occur when you have an IBM Cloud Pak for AIOps policy that triggers an IBM Tivoli Netcool/Impact policy when certain types of events are received. If this issue occurs and stops the event processing, the Impact connector logs or Impact policylogger logs can include messages that are similar to the following example exceptions:

```
[6/14/23, 11:38:45:816 UTC] 0000005d ConnectorMana W failed to send status update
...
[6/14/23, 11:38:45:815 UTC] 000023ca StandardConne W configuration stream terminated with an error
...
[6/14/23, 11:38:45:816 UTC] 000023cc GRPCCloudEven W consume stream terminated with an error:
channel=cp4waiops-cartridge.lifecycle.output.connector-requests
```

If you encounter this issue, you might need to restart the **impact-connector** pod to begin the processing of the event stream again.

IBM Tivoli Netcool/Impact stops sending data to widgets in a JazzSM dashboard

After Creating IBM Tivoli Netcool/Impact connections, the IBM Tivoli Netcool/Impact server stops sending data to widgets in a JazzSM dashboard. The widget will report an **ATKRST132E** or **404** error code.

Solution: Re-create the parameter files for the AIOPS_HandleAction and AIOPS_ExecJS Impact policies.

For more information, refer to the troubleshooting article Netcool/Impact datasets for DASH stop working after deploying the AIOps to Impact connector.

Ansible Automation Controller integration not running due to prohibited egress

After configuring an Ansible Automation Controller integration, the integration reports *Not running*, and no Ansible templates are shown in the *Automation Actions* table.

Use the following steps to check whether the failure is due to prohibited egress:

1. Run the following commands to view the RBA Automation Service pod logs, and check whether responses are being received from Ansible:

   ```
   export AIOPS_NAMESPACE=<AIOps installation namespace>
   oc logs -l app.kubernetes.io/component=rba-as -n ${AIOPS_NAMESPACE}
   ```

   Example output if responses are not being received from Ansible:

   ```
   Error: awxp/requestAwx.requestAwx: Request to Ansible failed... Request was not responded after 5
   seconds
   ```

2. Run the following commands to attempt to connect to the Ansible instance from the RBA Automation Service pod:

   ```
   export AIOPS_NAMESPACE=<AIOps installation namespace>
   oc exec $(oc get pod -l app.kubernetes.io/component=rba-as -n ${AIOPS_NAMESPACE} -o
   jsonpath='{.items[0].metadata.name}') -c rba-as -n ${AIOPS_NAMESPACE} -it -- /bin/bash

   # An interactive shell inside the container will open
   curl -k -vvv <Your Ansible host>
   ```

Solution: If responses are not received from Ansible and you cannot connect to Ansible, then create a NetworkPolicy that allows egress from the RBA service pods. Run the following commands:

```
export AIOPS_NAMESPACE=<AIOps installation namespace>

cat << EOF | oc apply -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: aiops-rba-rbs-egress
  namespace: ${AIOPS_NAMESPACE}
spec:
  egress:
  - {}
  podSelector:
    matchLabels:
      app.kubernetes.io/component: rba-rbs
  policyTypes:
  - Egress
EOF

cat << EOF | oc apply -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: aiops-rba-as-egress
```

```
  namespace: ${AIOPS_NAMESPACE}
spec:
  egress:
  - {}
  podSelector:
    matchLabels:
      app.kubernetes.io/component: rba-as
  policyTypes:
  - Egress
EOF
```

Troubleshooting Secure Tunnel

Learn how to isolate and resolve problems with Secure Tunnel.

- Secure Tunnel connector is not running after restart with Podman
- fuser hangs when Secure Tunnel install script is launched
- When creating a Secure Tunnel connection, the status remains in pending for a long time
- Slack is unable to verify the secure tunnel URL
- Secure Tunnel connector pod in crash loop state after installation on Red Hat OpenShift Container Platform cluster

Secure Tunnel connector is not running after a restart with Podman

This issue can occur when you install the Secure Tunnel connector to a host machine on which Podman is installed. When the host machine is rebooted and the Secure Tunnel connector is checked by using the **`podman ps -a`** command, the Secure Tunnel connector container does not display **`running`** status.

If this issue occurs, the **`podman-restart`** service must be activated by using the **`systemctl`** command:

```
systemctl start podman-restart
systemctl enable podman-restart
```

After entering the command, check **`podman-restart`** worked by using the following command:

```
systemctl status podman-restart
```

If the Connector is still not running, try restarting the host machine.

fuser hangs when Secure Tunnel install script is launched

After launching the Secure Tunnel install script, fuser hangs and you see something that resembles this

```
Checking whether docker or podman is installed ...
docker found.
Starting docker engine...
fuser found.
```

This issue can occur when fuser is having trouble finding a specific available port and so the execution stalls.

Solution: This issue rarely happens but should you encounter it edit the **`install-connector.sh`** and add a timeout in the fuser command.

Change **`PortCheck=$(fuser $HEALTHY_PORT/tcp 2>/dev/null)`** to **`PortCheck=$(timeout 2 fuser $HEALTHY_PORT/tcp 2>/dev/null)`**. This change can resolve the issue.

When creating a Secure Tunnel connection, the status remains in pending for a long time

When you use the **Custom** method to create a Secure Tunnel connection, the **Status** of the connection is stuck in **Pending** for a long time. This also occurs when you create a secure tunnel connection that uses a shortcut (for example, ChatOps, VMware or OpenStack).

This issue can occur when the **`tls secret`** was not created by the CertManager but the tunnel connection is waiting for it.

Solution: In most cases, re-creating the connection resolves the problem. If that does not work, you need to run the following command to restart the **IBM Cert Manager** and re-create the certificate CR to resolve this problem. Replace the namespace used here, **`cp4aiops`**, with your own namespace.

```
CONNECTION_NAME=<the name of the tunnel connection that you created above >
for NAME in $(oc -n ibm-common-services get pod --no-headers | grep cert-manager | awk '{print $1}')
do
```

```
    oc -n ibm-common-services delete pod ${NAME}
done
CONNECTION_ID=`oc -n cp4aiops get TunnelConnection ${CONNECTION_NAME} -o=jsonpath={.status.id}`
oc -n cp4aiops delete certificates.certmanager.k8s.io sre-tunnel-${CONNECTION_ID}-server-cert
oc -n cp4aiops delete certificates.certmanager.k8s.io sre-tunnel-${CONNECTION_ID}-connector-cert
oc -n cp4aiops delete certificates.certmanager.k8s.io sre-tunnel-${CONNECTION_ID}-api-cert
```

Slack is unable to verify the secure tunnel URL

You can encounter an issue with Slack being unable to verify the secure tunnel URL after you set up a ChatOps secure tunnel connection. This issue can occur even when the Secure Tunnel UI shows that the installation was successful.

This issue can occur if you enabled the Egress Network policy with the **`enableEgressNetworkPolicy: true`** when you install the Secure Tunnel.

You can use the following command to check whether the egress network policy is enabled. If the egress network policy is enabled the output should be **`true`**.

```
oc -n <the namespace of the aiops> get Tunnel sre-tunnel -o=jsonpath='{.spec.enableEgressNetworkPolicy}'
```

Solution: Use the following YAML code to help you create an egress network policy in your IBM Cloud Pak for AIOps project (namespace) to allow Secure Tunnel access to the **`ibm-nginx`** gateway.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: chatopstunnel-egress-ibm-nginx
  namespace: <the namespace of the aiops>
spec:
  podSelector:
    matchLabels:
      app.kubernetes.io/instance: sre-tunnel
      app.kubernetes.io/managed-by: tunnel
      app.kubernetes.io/name: sre-tunnel-chatopstunnel.app
      tunnel_connection_name: chatopstunnel
  egress:
    - to:
        - podSelector:
            matchLabels:
              app.kubernetes.io/component: ibm-nginx
              icpdsupport/app: gateway
              component: ibm-nginx
  policyTypes:
    - Egress
```

Secure Tunnel connector pod in crash loop state after installation on Red Hat OpenShift Container Platform cluster

You might notice an issue where the Secure Tunnel connector pods stay in a crash loop state. The pod logs might show proxy related errors.

This issue can occur when the Secure Tunnel connector is installed on a cluster within an air-gapped (offline) environment that does not have outbound network access.

Solution: For the Secure Tunnel connector that is installed within the air-gapped (offline) environment, create a network connection with the Secure Tunnel server by using an HTTP proxy. You can create the connection by completing the following steps:

1. From a command line, go to the **`tunnel-connector-install-scripts`** directory.
2. Open the **`tunnel-connector.yaml`** file by using the following command:

   ```
   vi tunnel-connector.yaml
   ```

   When you open the YAML file, it can resemble the following example YAML file:

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     labels:
       app.kubernetes.io/name: sre-tunnel-df012501eede4b02.secret
       app.kubernetes.io/managed-by: tunnel
       app.kubernetes.io/instance: sre-tunnel
       tunnel-network-name: woprod-test
       tunnel_connection_name: woprod-test
       tunnel_connection_id: "df012501eede4b02"
     name: sre-tunnel-df012501eede4b02-config
   data:
   ```

```
  config_connector.yaml: |
    <base64 encoded data>
type: Opaque
```

3. Decode the **<base64 encoded data>** of the YAML file by using the following command:

   ```
   echo <base64 encoded data> | base64 -d
   ```

   Where **<base64 encoded data>** is the encoded data from the YAML file.

4. Update the **websocket_http_proxy** and **websocket_https_proxy** properties that are found in the decoded data.

   ```
   websocket_http_proxy: <http proxy endpoint>
   websocket_https_proxy: <https proxy endpoint>
   ```

   Where

   - **<http proxy endpoint>** is the HTTP proxy endpoint. This proxy is used by the tunnel connector to create a nonencrypted network web socket connection with the tunnel server. For example, the value can be **proxy.example.ibm.com:3128**.
   - **<https proxy endpoint>** is the HTTPS proxy endpoint. This proxy is used by the tunnel connector to create an encrypted network web socket connection with the tunnel server. For example, the value can be **proxysecure.example.ibm.com:3128**.

5. Encode the data by using the following command.

   ```
   echo '<decoded data>' | base64
   ```

   Where **<decoded data>** is the decoded data with the two modified properties.

6. Add the Base64 encoded data to the **<base64 encoded data>**.

   ```
   cat <<EOF | oc -n <namespace> apply -f -
   apiVersion: v1
   kind: Secret
   metadata:
     labels:
       app.kubernetes.io/name: sre-tunnel-df012501eede4b02.secret
       app.kubernetes.io/managed-by: tunnel
       app.kubernetes.io/instance: sre-tunnel
       tunnel-network-name: woprod-test
       tunnel_connection_name: woprod-test
       tunnel_connection_id: "df012501eede4b02"
     name: sre-tunnel-df012501eede4b02-config
   data:
     config_connector.yaml: |
       <base64 encoded data>
   ```

   ```
   type: Opaque
   EOF
   ```

   Where

   - **<base64 encoded data>** is the Base64 encoded data.
   - **<namespace>** is the namespace where Secure Tunnel connector is installed.

7. Restart the Secure Tunnel connector pods by using the following command:

   ```
   oc delete pod <secure-tunnel-connector-pod> -n <namespace>
   ```

   Where

   - **<secure-tunnel-connector-pod>** is the name of the Secure Tunnel connector pod.
   - **<namespace>** is the namespace where Secure Tunnel connector is installed.

Troubleshooting AI model management

Learn how to isolate and resolve problems with AI model training.

- Larger data sets are taking a long time to train
- Log anomaly detection training is taking longer than expected
- Network issues with Elasticsearch
- Network issues with Spark
- Training encounters insufficient resources
- Training reports failure but process completes
- Log anomaly detection model deployment fails with error undefined
- Ingested raw logs are not available
- Statistical baseline reference can require manual reset

- How to stop a precheck job
- Evaluation of historical metric data gives inconsistent results
- Model training considers less historical data in a data quality check
- Deployed model status in a metric anomaly or temporal grouping training algorithm is changed to pending
- Model version deployment reports failure despite being successfully deployed
- Metric anomalies are not detected
- The Alert suppression XinY policies tile does not appear in AI model management

Larger data sets are taking a long time to train

If your data sets are large, it can take a long time for AI training to complete. For example, when you run log anomaly training a large log data set might take a full day to train.

By default, two Luigi pods are assigned to handle training jobs. This number of pods is sufficient to complete training within acceptable time periods if your data sets are not too large. However, larger data sets can take a considerable amount of time to train based on two Luigi pods that are running in parallel.

Solution: To speed up training, you must increase the number of Luigi pods that are dedicated to training. The default value is two pods, regardless of the type of training definition that you are running.

A limit applies to the number of Luigi pods that can be dedicated to training. Before you increase the number of Luigi pods that are dedicated to training, it is important for you to be aware of these limits.

Each additional Luigi pod requires approximately the following extra resources:

- 2.5 CPU cores
- 2 - 16 GBytes of memory, depending on the AI type that is trained.

Complete the following steps:

1. Log in to your cluster.
2. Go to the target AIOps namespace.
3. Find the Luigi Resource Quota by running the following command:

   ```
   oc get resourcequota | grep luigi
   ```

4. Edit the resource quota by running `oc edit resourcequota <name of resource quota found in step 3>`. Also, edit the `spec.hard.pods` field with the number of pods you want to dedicate to training simultaneously. Ensure that you have the needed resources as described in the Before you begin section.
5. Save the edit by typing `:wq` or `:x`. After a few minutes, the changes are automatically applied to your cluster, and you can start training.

Log anomaly detection training is taking longer than expected

Log anomaly detection training is taking longer than expected. Currently, log training times are on the order of 860 log records per second. This number is significantly slower than the 8,000 log records per second that is expected in enterprise environments. Log training time is proportional to the volume of log data passed through the training system.

This issue can occur as the training rate is constrained by the current system architecture.

Solution: This issue has no simple resolution. Depending on the data you are training, the following actions might have some impact on training times:

- Increase the number of Luigi pods that are dedicated to training, as described in the **Solution** section of Larger data sets are taking a long time to train.
- Decrease the sampling size used for the training.

Network issues with Elasticsearch

The IBM Cloud Pak for AIOps IBM Cloud Pak Automation console is generating lots of network errors.

This issue occurs when Elasticsearch is in an unhealthy state. It occurs most likely due to issues with Elasticsearch connectivity.

Solution: Try to get more information on potential Elasticsearch issues by examining the AI Platform pod logs. Check to see whether any error logs associated with Elasticsearch connectivity exists.

1. Log in to your cluster by running the `oc login` command. You must be an administrator user within the project (namespace) where IBM Cloud Pak for AIOps is installed.

```
oc login -u <username> -p <password>
```

For more information, see Logging in to the OpenShift CLI.

2. Display the logs and look for error logs associated with Elasticsearch connectivity.

```
oc logs $(oc get pods | grep ai-platform-api-server| awk '{print $1}')
```

Network issues with Spark

The AI platform has problems either launching training jobs, or reporting ongoing training status for the temporal grouping AI algorithm. As a result, you encounter errors on the Manage AI algorithms page.

This issue can occur as training jobs for the temporal grouping AI algorithm are orchestrated by the internal component Spark. When there are issues with launching training jobs or training status is not being displayed, it indicates that Spark is in an unhealthy state.

Solution: Try to get more information on potential Spark issues by examining the AI Platform pod logs.

Proceed as follows:

1. Log in to your cluster by running the **`oc login`** command. You must be an administrator user within the project (namespace) where IBM Cloud Pak for AIOps is installed.

```
oc login -u <username> -p <password>
```

For more information, see Logging in to the OpenShift CLI.

2. Display the logs and look for error logs associated with Spark.

```
oc logs $(oc get pods | grep ai-platform-api-server| awk '{print $1}')
```

Training encounters insufficient resources

Training an AI algorithm returns an insufficient resources exception. The reason for this exception is that internal components within the training process were unable to locate a node with sufficient CPU or memory requirements to satisfy the training run.

Solution: The solution is to reduce the number of CPU and memory requests. Usually memory is the bottleneck so if you are unsure of what to do, a good option is to initially reduce memory by half.

1. Find the custom resource of the algorithm for which you want to alter the default resource requirements by running the following command:

```
oc get algorithms
```

2. On running the command, custom resources containing all the algorithm specifications are displayed as follows:

```
oc get algorithms
```

```
NAME                AGE
change-risk         46h
log-anomaly         46h
similar-incidents   46h
```

3. Edit the specific algorithm by running the following command:

```
oc edit algorithm <name>
```

4. Decode the *spec.manifestBase64* field. Once decoded, you see a *resources* field that is defined as follows:

```
resources:
    requests:
        cpu: 2
        memory: 4Gi
    limits:
        cpu: 2
        memory: 4Gi
```

Note: You can edit requests and limits (make sure that **limits=requests** for training algorithms), re-encode to base64, and then replace the *manifestBase64* field in the CR with this newly encoded manifest.

5. Save the changes, and proceed with your training.

Training reports failure but process completes

An AI algorithm training run has completed, but the IBM Cloud Pak Automation console shows that the training has failed. No concrete model was generated during the training run.

The reason for this failure might be insufficient data or a misconfiguration in the range of the data set. Failure does not mean that the training failed; it just means that no value was obtained from the training run and consequently no model was produced.

Solution: Examine the data sources for the training, examine the training configuration, make any necessary changes, and launch the training again.

Log anomaly detection model deployment fails with error undefined

When you are training the log anomaly detection algorithm with historical data and you attempt to deploy the AI model, you might encounter an issue where the model deployment fails with an **`Error: undefined`** error in the UI.

Solution: As a workaround when this issue occurs, restart all **`cp4waiops-eventprocessor`** pods, which can cause the model deployment to complete successfully. To restart the pods, run the following command, which deletes the current instance of the pods and creates and starts a new instance of the pods.

```
oc delete po cp4waiops-eventprocessor-eve-29ee-ep-jobmanager-0 cp4waiops-eventprocessor-eve-29ee-ep-jobmanager-1 cp4waiops-eventprocessor-eve-29ee-ep-taskmanager-0 cp4waiops-eventprocessor-eve-29ee-ep-taskmanager-1
```

Note: If you installed IBM Cloud Pak for AIOps with a larger production deployment sizing, you need to specify any additional **`cp4waiops-eventprocessor`** pods that exist when you run the preceding command.

Ingested raw logs are not available immediately

Ingested raw logs are not available immediately for "Log anomaly detection - natural language" model training after the import is complete. Raw logs are not available for review in the "Preview log" section of the ChatOps Alert Viewer.

This issue can occur as additional time is required to store normalized logs in Elasticsearch after they are ingested. The amount of time can be reduced by allocating more Flink tasks to the Flink job that stores the normalized logs.

Solution: To resolve this issue, complete the following steps:

1. The environment variables that control the number of Flink tasks that are used to move logs to Elasticsearch are located in the Kubernetes deployment for aimanager-aio-controller. Edit this deployment: oc edit deployment aimanager-aio-controller
2. Find the environment variables by searching for "LOGS_JOB2_BASE_PARALLELISM". Change the value of LOGS_JOB2_BASE_PARALLELISM and LOGS_JOB2_NUM_PARTITIONS from their default values of 2 to something higher. Consider doubling the value to see whether it has the wanted effect.
3. Save the yaml and wait for the pods to restart. You can verify that the pods have restarted by monitoring the output of: oc get pods | grep aimanager-aio-controller
4. Stop the data flow for all logs connections. It is important that there are no active logs connections, as this is how the controller knows that it is OK to stop the Flink job, which copies the logs to Elasticsearch.
5. Restart the data flow for all logs connections.

The Flink job restarts by using a higher number of tasks. It copies the logs to Elasticsearch more quickly.

Be careful not to allocate a higher parallelism than you have Flink task slots available. The number of tasks that you need can be calculated by adding the following values:

- sum of base_parallelism from all defined connections, as defined in the connection UI
- value of LOGS_JOB3_BASE_PARALLELISM multiplied by the number of logs type defined connections (for example, LogDNA, Falcon Logscale, Splunk, and so on)
- value of LOGS_JOB2_BASE_PARALLELISM
- value of LOGS_JOB4_BASE_PARALLELISM

If you need additional task manager slots, you can add a task manager by Increasing data streaming capacity.

Statistical baseline reference can require manual reset

There is a chance that infrequently occurring anomalies are learned as normal behavior and they might not be detected anymore. For instance, if interval for the infrequently occurring anomalies do not exceed the predefined threshold of 0.15, the interval is not tagged as biased and the reference values are updated and the anomaly is learned as normal behavior. If this occurs, you need to reset the learning by resetting the reference of the statistical baseline with the following steps:

1. Open a terminal and log in to your cluster:

   ```
   oc login <server> -u <cluster username> -p <cluster pass>
   ```

2. Open another terminal and run port-forwarding for Elasticsearch

   ```
   while true; do oc port-forward $(oc get po | grep iaf-system-elasticsearch-es-aiops-0 | awk '{print $1}') 9200; done
   ```

3. Get the Elasticsearch credentials:

```
export username=$(oc get secret iaf-system-elasticsearch-es-default-user -o jsonpath="
{.data.username}"|base64 -d);
export password=$(oc get secret iaf-system-elasticsearch-es-default-user -o jsonpath="
{.data.password}"|base64 -d);
```

4. Reset the statistical baseline reference:

```
curl -u $username:$password -XDELETE https://localhost:9200/1000-1000-reference_oob --insecure | sort
```

How to stop a precheck job

The workaround is to stop the precheck job manually using Openshift CLI commands, delete the training definition, recreate the training definition, and relaunch the training configuration with the correct details.

There are two methods to do this. The first is recommended as it limits mistakes and there is less risk of making training off sync with data in the Elasticsearch database, but the models are deleted if you use this method, whereas with the second method that is listed they will not, so it's preferable to use the second method if you want to conserve the models in question.

Method 1: Stop the precheck job and delete the training definition to start afresh

1. Retrieve the precheck job in the IBM Cloud Pak for AIOps namespace, or a different namespace that it is installed under:

```
oc get jobs | grep precheck
```

2. Delete the precheck job in order to stop the precheck run:

```
oc delete job <job-name>
```

3. Delete the training definition using the delete button on the console.
4. Re-create the training definition with the correct configurations required for the training algorithm, for example date range or scheduling.
5. Run the training as usual.

Method 2: Stop precheck job and clean up Elasticsearch data

1. Retrieve the precheck job in the IBM Cloud Pak for AIOps namespace, or a different namespace that it is installed under:

```
oc get jobs | grep precheck
```

2. Delete the precheck job in order to stop the precheck run:

```
oc delete job <job-name>
```

3. Retrieve the AI Platform pod name in the IBM Cloud Pak for AIOps namespace or different namespace that it is installed under:

```
oc get pod | grep api-server
```

4. Enter inside the pod:

```
oc rsh <pod-name> # Example: oc rsh aimanager-aio-ai-platform-api-server-55d5b59bd4-jk7nz
```

5. Verify the latest precheck run and retrieve the '_id' required for cleanup:

```
curl -k -u <username>:<password> -X POST $ES_URL/precheckrun/_search?pretty=true" -H 'Content-Type:
application/json' -d '{"query": {"match_all": {}},"size": 1,"sort": [{"timestamp": {"order": "desc"}}]}'
```

Example:

```
"_id":
"log_anomaly_detection_configurationv3"
```

6. Use the '_id' and version specified in the ID to delete the precheck respective run entries in elasticsearch:

```
curl -k -u <username>:<password> -X POST $ES_URL/precheckrun/_delete_by_query" -H 'Content-Type:
application/json' -d '{"query": {"term": {"_id": <ID retrieved from previous step>}}}'
```

Example:

```
curl -k -u <username>:<password> -X POST $ES_URL/precheckrun/_delete_by_query" -H 'Content-Type:
application/json' -d '{"query": {"term": {"_id": "log_anomaly_detection_configurationv3"}}}'
```

```
curl -k -u <username>:<password> -X POST $ES_URL/precheckstatus/_delete_by_query" -H 'Content-Type:
application/json' -d '{"query": {"term": {"precheckRunVersion": <version retrieved from previous
step>}}}'
```

Example:

```
curl -k -u <username>:<password> -X POST $ES_URL/precheckstatus/_delete_by_query" -H 'Content-Type:
application/json' -d '{"query": {"term": {"precheckRunVersion": "v3"}}}'
```

7. Update the training definition document to decrement the precheck run version:

```
curl -k -u <username>:<password> -X POST
$ES_URL/trainingdefinition/_update/log_anomaly_detection_configuration" -H 'Content-Type:
application/json' -d '{"doc": {"precheckRunVersion": <decremented version>}}'
```

Example:

```
curl -k -u <username>:<password> -X POST
$ES_URL/trainingdefinition/_update/log_anomaly_detection_configuration" -H 'Content-Type:
application/json' -d '{"doc": {"precheckRunVersion": "v2"}}'
```

8. Make your changes to the training configuration, for example to the date range or scheduling.
9. Run the training as usual.

Evaluation of historical metric data gives inconsistent results

Sending historical metric data for training – such as in a product demonstration – you see inconsistent outcomes in the IBM Cloud Pak Automation console metric anomaly results.

At inference (metric evaluation) time, it takes a short amount of time to load analytics models into memory. This delay might lead to some metric data being left out of an evaluation.

Solution: Send in one datapoint, for each individual metric, to allow the model to be loaded. Then, send remaining data after a short interval, 30 - 60 seconds later.

Model training considers less historical data in a data quality check

Less data is included in a data quality check if you select **Training** first instead of running a precheck and training separately.

If you select **Training** to run both a data quality check and training sequentially, you might notice this problem. This situation occurs only when some connections are moved into live data after the training completes, and the training dates are changed to check historical data from another connection source.

Solution: Run the precheck and training separately. Start the data quality precheck, and after the precheck is complete, run the training after.

Deployed model status in a metric anomaly or temporal grouping training algorithm is changed to pending

A temporal grouping or a metric anomaly detection algorithm is in a pending state although the model was previously trained and deployed with success.

Solution: To resolve this issue, complete the following steps:

1. Run the `oc -n <namespace> rsh <aimanager-aio-ai-platform-api-server-*>` command.
2. Delete all versions of the `metric_anomaly_detection_configuration` or `temporal_grouping_configuration` training algorithms in the `trainingrun` index. Search the index by running the following command:

```
./elastic.sh -X GET -E trainingrun/_search\?pretty=true&size=1000
```

3. Identify all the documents that are related to your `metric_anomaly_detection_configuration` or `temporal_grouping_configuration` training algorithms. The `_id` field of the documents is `metric_anomaly_detection_configurationv<version>` or `temporal_grouping_configurationv<version>`, where `<version>` is the version of the training run.
4. Delete the training run documents:

```
./elastic.sh -X DELETE -E trainingrun/_doc/metric_anomaly_detection_configurationv1
./elastic.sh -X DELETE -E trainingrun/_doc/metric_anomaly_detection_configurationv2
[..]
```

or

```
./elastic.sh -X DELETE -E trainingrun/_doc/temporal_grouping_configurationv1
./elastic.sh -X DELETE -E trainingrun/_doc/temporal_grouping_configurationv2
[..]
```

5. Delete the training definition to clean the UI:

   ```
   ./elastic.sh -X DELETE -E trainingdefinition/_doc/metric_anomaly_detection_configuration
   ```

   or

   ```
   ./elastic.sh -X DELETE -E trainingdefinition/_doc/temporal_grouping_configuration
   ```

6. Now, you can train a new model.

Model version deployment reports failure despite being successfully deployed

When you deploy LAD model version manually, you might encounter an error message on your console that reports deployment failure. The error message reads that the failure occurs due to the version being already deployed.

This error arises due to the successful deployment of the model version in the background despite getting timed out.

Solution: Restart the data flow for all the connections. To do so, first disable and then re-enable the data flow for all connections.

Metric anomalies are not detected

Golden signals log anomalies are not detected after you upgrade to IBM Cloud Pak for AIOps version 4.2.1.

Solution: Resave the log connections on the AI Model Management UI.

The Alert suppression XinY policies tile does not appear in AI Model Management

In the IBM Cloud Pak Automation console, click **AI model management** and find the *Alert suppression XinY policies* tile. If this tile does not appear, an administrator or developer user needs to restart the **`aimanager-aio-ai-platform-api-server`** pod.

To restart the target pod, delete the pod so that a new pod automatically starts up:

1. Connect to your cluster with the **`oc login`** command.
2. Restart the **`aimanager-aio-ai-platform-api-server`** pod.

   ```
   oc delete pod/"$(oc get pod|grep aimanager-aio-ai-platform-api-server|awk '{print$1; exit}')"
   ```

   Successfully running this command produces output similar to the following example:

   ```
   pod "aimanager-aio-ai-platform-api-server-55c5d975c4-6qpwk" deleted
   ```

Troubleshooting alert management

Learn how to isolate and resolve problems with alert management.

- No alert or incident created when an event is sent
- No alert ranking with large installations
- Error message "Reached the max retries per request limit (which is 20)" is displayed when viewing the Incidents and Alerts page
- Closed alerts are not being removed from Alert Viewer

No alert or incident created when an event is sent

Symptom A: An event is sent, but no alert is received by IBM Cloud Pak® for AIOps and no incident is created.

Running **`waiops-mustgather.sh -V flink-jobs`** using the must-gather tool, returns **`"RESTARTING"`** or **`"FAILED"`** under the entry **`[FLINK = aiops-ir-lifecycle]`**. For example:

```
[FLINK = cp4aiops]
{
  "jobs": []
}

[FLINK = aiops-ir-lifecycle]
{
  "jobs": [
    {
      "id": "4e442c7880239bede340e953e6b920b3",
      "status": "RESTARTING"
```

```
    }
  ]
}
```

This issue might be caused by a volume of event data that is too large for the given installation size and configured policy set. This can cause the policy engine to run out of available memory.

Solution: To resolve this issue, complete the following steps:

1. Evaluate the sizing of the installation compared to the current event load. If there is a miss-match, change the installed sizing, or the configured event sources.
2. Ensure that incident creation policies are only configured to raise incidents for alerts which require immediate resolution. Incidents are more resource intensive to process than alerts, so creating incidents for all alerts will require a larger sized installation.
3. After re-configuring the system, the policy engine may still be in a failure state as it tries to re-process the already-inserted events. It is possible to reset the policy engine so that it only processes new events using the must-gather tool:

   ```
   waiops-mustgather.sh -DR -C clear-lifecycle-state.sh
   ```

   NOTE: Resetting the policy engine will cause all policy state to be lost. This may result in events creating new alerts and incidents rather than being deduplicated into existing ones. It will also mean that any events inserted in the time between the failure and running this command will not be processed by the system.

4. Restart the `flink` pods:

   1. Search for the pods that match the string `aiops-ir-lifecycle-eventprocessor-ep`
      This should return four pods.
   2. Right-click and **Delete** the pods.
      The pods should restart immediately.

Symptom B: An event is sent, but no alert is received by IBM Cloud Pak® for AIOps and no incident is created. The log of the lifecycle task manager pod contains the following error:

```
WARNING: Exception thrown during asynchronous load
java.util.concurrent.CompletionException: io.grpc.StatusRuntimeException: UNAVAILABLE: ssl exception
```

The issue might be a certificate mismatch between the lifecycle operator and the policy GRPC server, causing the connection to fail with SSL exceptions.

Solution: To resolve this issue, complete the following steps:

1. Run the following command to delete the certificate:

   ```
   oc delete secret aiops-ir-lifecycle-policy-grpc-clt-svc aiops-ir-lifecycle-policy-grpc-svc
   ```

2. Get the lifecycle-operator pod name:

   ```
   oc get pods |grep ir-lifecycle-operator-controller-manager
   ```

3. Restart the lifecycle operator:

   ```
   oc delete pod ir-lifecycle-operator-controller-manager-<lifecycle-operator-pod-name>
   ```

No alert ranking with large installations

On larger production scale installations of IBM Cloud Pak® for AIOps, no ranking of alerts is displayed in the Alert Viewer.

Solution: To resolve this issue, complete the following steps:

1. Run the following command to find the `classifier` and `probablecause` pods:

   ```
   oc get pods | egrep -i "classifier|probablecause"
   ```

   For example:

   ```
   oc get pods | egrep -i "classifier|probablecause"

   aiops-ir-analytics-classifier-6477fd7788-4rzwx                    1/1     Running     0
   aiops-ir-analytics-classifier-6477fd7788-8jd55                    1/1     Running     0
   aiops-ir-analytics-probablecause-5b89d56c44-b752h                 1/1     Running     1
   aiops-ir-analytics-probablecause-5b89d56c44-kpqcb                 1/1     Running     1
   ```

2. Restart the pods:

   ```
   oc delete pods aiops-ir-analytics-classifier-6477fd7788-4rzwx aiops-ir-analytics-classifier-6477fd7788-
   8jd55 aiops-ir-analytics-probablecause-5b89d56c44-b752h aiops-ir-analytics-probablecause-5b89d56c44-
   ```

```
kpqcb
```

Error message "Reached the max retries per request limit (which is 20)" is displayed when viewing the Incidents and Alerts page

The **redis-m** and **cem-users** pods are not ready, and the **Redis** logs have errors similar to the following:

```
ERROR: redis/25 on aiops-ir-core-cem-users-5b4497dd6-77wdc: Error from redis client
MaxRetriesPerRequestError: Reached the max retries per request limit (which is 20).
```

```
Bad file format reading the append only file: make a backup of your AOF file, then use ./redis-check-aof --
fix <filename>
```

Solution: **Redis** files are corrupted. Use the following procedure to fix this.

1. Find the names of your **Redis** member pods.

   ```
   oc get pods | grep redis-m
   ```

   Example output:

   ```
   oc get pods | grep redis-m

   c-example-redis-m-0   4/4     Running     0     4h54m
   c-example-redis-m-1   3/4     Running     0     139m
   c-example-redis-m-2   3/4     Running     0     136m
   ```

2. For each of the pods returned in step 1, run the following steps. It is important that you run this step for each of the **Redis** member pods, even if they are showing as ready.

   1. Exec into the **Redis** pod and use the **mgmt** container, which has the necessary **redis-check-aof** tool.

      Example command:

      ```
      oc exec -it c-example-redis-m-0 -c mgmt bash
      ```

   2. Fix the corrupted file in the **Redis** pod.

      ```
      # Take a backup of the corrupted AOF file
      cp /data/appendonly.aof /data/appendonly.aof.bkp

      # Fix the corrupted AOF file
      ./usr/local/bin/redis-check-aof --fix /data/appendonly.aof

      # Exit the container
      exit
      ```

      Important: You must run steps a and b for each of the **Redis** pods returned in step 1.

Closed alerts are not being removed from Alert Viewer

You might notice an issue where closed alerts are not being removed from the Alert Viewer.

If this issue occurs, you might see an error that is related to timeouts for requests to Postgres. This error can be similar to the following example:

```
[2023-12-12T17:30:30.022Z] ERROR: datalayer.postgres/14 on aiops-ir-core-ncodl-jobmgr-7f4c64dcdd-d7xjz: Error
detected on streaming query
    Error: Query timed out
        at Timeout._onTimeout (/app/lib/helpers/dbservice/lib/helpers/dbservice/postgres.js:178:20)
        at listOnTimeout (node:internal/timers:559:17)
        at processTimers (node:internal/timers:502:7)
[2023-12-12T17:30:30.022Z] ERROR: datalayer.postgres/14 on aiops-ir-core-ncodl-jobmgr-7f4c64dcdd-d7xjz: Error
detected on client connection. Ending query stream (reqid=13778660-9914-11ee-844d-e573891ebc4b)
    Error: Query timed out
        at Timeout._onTimeout (/app/lib/helpers/dbservice/lib/helpers/dbservice/postgres.js:178:20)
        at listOnTimeout (node:internal/timers:559:17)
        at processTimers (node:internal/timers:502:7)
[2023-12-12T17:30:30.024Z] ERROR: datalayer.postgres/14 on aiops-ir-core-ncodl-jobmgr-7f4c64dcdd-d7xjz: Error
detected on streaming query cursor
    Error: Connection terminated
        at Connection.<anonymous> (/app/node_modules/pg/lib/client.js:132:36)
```

```
        at Object.onceWrapper (node:events:627:28)
        at Connection.emit (node:events:525:35)
        at Socket.<anonymous> (/app/node_modules/pg/lib/connection.js:63:12)
        at Socket.emit (node:events:525:35)
        at TCP.<anonymous> (node:net:301:12)
```

This issue might be caused by the failure of the datalayer job manager. Since the job is not running, there is no automated process for clearing the closed alerts.

Solution: Use the following steps to resolve the closed alerts issue:

1. Determine the name of the datalayer job manager pod.

   ```
   oc get pod -l "app.kubernetes.io/component=ncodl-jobmgr"
   ```

   The output from the preceding command can look like the following example::

   ```
   [root@6c025b05c0a7 /]# oc get pod -l "app.kubernetes.io/component=ncodl-jobmgr"
   NAME                                      READY   STATUS    RESTARTS   AGE
   aiops-ir-core-ncodl-jobmgr-85458b4979-gqgj8   1/1     Running   0          5h11m
   ```

2. Use the pod name to describe the job manager pod. If the pod is in a crash loop or not scheduled status, use the following command to see the failure:

   ```
   oc describe pod <pod name>
   ```

   Where **<pod name>** is the name of the datalayer job manager pod.

   If the failure reason from describe command is not clear or the pod appears to be running fine then check the pod logs for potential run failure conditions. Use the following command to check the logs:

   ```
   oc logs <pod name>
   ```

   Where **<pod name>** is the name of the datalayer job manager pod.

   You can, for example, search for "Alert delete job" in the logs. "Alert delete job" must be running. The logs for delete job are shown in the following example.

   ```
   [2023-12-12T17:38:00.002Z]  INFO: datalayer.jobmgr/14 on aiops-ir-core-ncodl-jobmgr-7f4c64dcdd-d7xjz:
   Running: Alert delete job [2023-12-12T17:38:30.120Z]  INFO: datalayer.jobmgr/14 on aiops-ir-core-ncodl-
   jobmgr-7f4c64dcdd-d7xjz: Complete: Alert delete job
    metrics: {
      "fetched": 0,
      "deleted": 0,
      "failed": 0
    }
    --
    psql: [
      {
        "name": "deletes",
        "mean": 0,
        "min": 0,
        "max": 0
      },
      {
        "name": "relatedDeletes",
        "mean": 0,
        "min": 0,
        "max": 0
      }
    ]
   ```

3. Use the following command to check whether the postgres instance is **Running**:

   ```
   oc get pods | grep postgres
   ```

   The postgres pod name might be similar to **ibm-cp-watson-aiops-edb-postgres-***. The asterisk (*) represents multiple pods that might be running for postgres.

4. If the Postgres instance is running fine, then it is possible that the instance might be busy and overloaded. If you see the preceding error such as "timeouts on request Postgres" or if the Postgres instance is running fine, in both cases use the following script to manually purge and recover the system.

   ```
   #!/bin/bash

   EDBPOD=ibm-cp-watson-aiops-edb-postgres-1
   EBDSECRET=aiops-ir-core-postgresql
   ```

```
# Get the credentials from required sec
USERNAME=$(oc get secret ${EBDSECRET} -o=jsonpath='{.data.username}' | base64 -d)
PASSWORD=$(oc get secret ${EBDSECRET} -o=jsonpath='{.data.password}' | base64 -d)
DBNAME=$(oc get secret ${EBDSECRET} -o=jsonpath='{.data.dbname}' | base64 -d)
SCHEMA=$(oc get secret ${EBDSECRET} -o=jsonpath='{.data.schema}' | base64 -d)

# RSH into EDB container.
oc rsh ${EDBPOD} psql -h localhost -p 5432 -U ${USERNAME} "dbname=${DBNAME} sslmode=require" -- <<EOF
${PASSWORD}

DELETE FROM aiops_irb.alerts WHERE state=2;
DELETE FROM aiops_irb.insights insight WHERE NOT EXISTS(SELECT 1 FROM aiops_irb.alerts alert WHERE
insight.parentid = alert.alertid) AND NOT EXISTS(SELECT 1 FROM aiops_irb.stories story WHERE
insight.parentid = story.storyid);
DELETE FROM aiops_irb.links link WHERE NOT EXISTS(SELECT 1 FROM aiops_irb.alerts alert WHERE
link.parentid = alert.alertid) AND NOT EXISTS(SELECT 1 FROM aiops_irb.stories story WHERE link.parentid
= story.storyid);
DELETE FROM aiops_irb.timeline timeline WHERE NOT EXISTS(SELECT 1 FROM aiops_irb.alerts alert WHERE
timeline.parentid = alert.alertid) AND NOT EXISTS(SELECT 1 FROM aiops_irb.stories story WHERE
timeline.parentid = story.storyid);
DELETE FROM relationships relation WHERE NOT EXISTS(SELECT 1 FROM aiops_irb.alerts alert WHERE
relation.childid = alert.alertid) AND NOT EXISTS(SELECT 1 FROM aiops_irb.stories story WHERE
relation.parentid = story.storyid);
DELETE FROM aiops_irb.alerts_details detail WHERE NOT EXISTS(SELECT 1 FROM aiops_irb.alerts alert WHERE
detail.identifier = alert.identifier);
EOF
```

Where **EDBPOD** is the Postgres pod name that you obtained previously.

The script deletes all closed alerts from the Alert Viewer. The job manager will also be able to remove the future closed alerts automatically.

A sample output from successful execution of the script:

```
Defaulted container "postgres" out of: postgres, bootstrap-controller (init)
Password for user aiops_irb:
DELETE 0
DELETE 0
DELETE 0
DELETE 0
DELETE 0
DELETE 0
```

The number after the **DELETE** keyword is the number of rows that are deleted.

Troubleshooting ChatOps

Learn how to isolate and resolve problems with ChatOps IBM Cloud Pak® for AIOps.

- Slack buttons are unresponsive
- Log preview dialog or View alerts dialog hangs intermittently in an incident in Slack or Microsoft Teams
- Alert and incidents do not appear in Slack
- Slack ChatOps connection is not working within an air-gapped (offline) environment
- Microsoft Teams ChatOps connection shows no response in the console
- During a ChatOps secure tunnel creation an 'installation failed' message displays

Slack buttons are unresponsive

Unresponsive buttons might be caused by a number of issues. If you encounter this issue, try completing the following steps to find the cause of the problem:

1. **Ensure two-way communication is established:**
    1. **Test two-way communication:** Incoming incidents are proof that one-way communication from IBM Cloud Pak for AIOps to Slack is successful. To check two-way communication, tag the IBM Cloud Pak for AIOps bot in the Slack channel. It responds with "Hey there I'm up and running." If it doesn't, IBM Cloud Pak for AIOps might not be receiving any communication from Slack.
    2. **Verify endpoints:** If the IBM Cloud Pak for AIOps bot did not respond, it might be that Slack is trying to communicate with the wrong endpoints, or the endpoints are not publicly accessible. To make sure that the endpoints are correct, open up the slack api for the app and navigate to the app manifest. The **event_subscriptions** and **interactivity** urls should match the ones that are produced by the IBM Cloud Pak for AIOps slack configuration and should be verified by Slack. If they do match but Slack has not verified them, that means Slack cannot access the endpoints likely because they are private. Slack needs some kind of proxy to access the private endpoints. A secure tunnel can be used in scenarios like this. For more information, see Secure Tunnel.

2. **Check communication speed:** If there are no problems with the steps that are outlined above, then it is possible the communication between Slack and IBM Cloud Pak for AIOps is taking too long. If a button is clicked, Slack waits 3 seconds to receive a response. If it doesn't get one, the communication is rejected and the button click fails. This is a rare occurrence, but can happen if the network communication is too slow, or if something else is preventing communication from happening in a timely manner. Try reconnecting and reconfiguring the Slack connection. For more information, see Creating a Slack connection. Then, verify that the issue is resolved.

3. **Look for errors in the ChatOps pods:** If 2-way comms are successful, and network communication seems to be fine, try logging in to the cluster to look at the logs. Look at the **`aimanager-aio-chatops-slack-integrator`** and **`aimanager-aio-chatops-orchestrator`** pods for any unusual errors and then contact the L2 support team.

Log preview modal or View alerts modal hangs intermittently in an incident in Slack or Microsoft Teams

This issue might happen when the Log Anomaly detection is in a high load, and takes longer time than normal to respond. When it occurs, you might encounter the following :

- When you select **View alerts** in an incident in Slack or Microsoft Teams, it hangs with the **`Loading ...`** message.

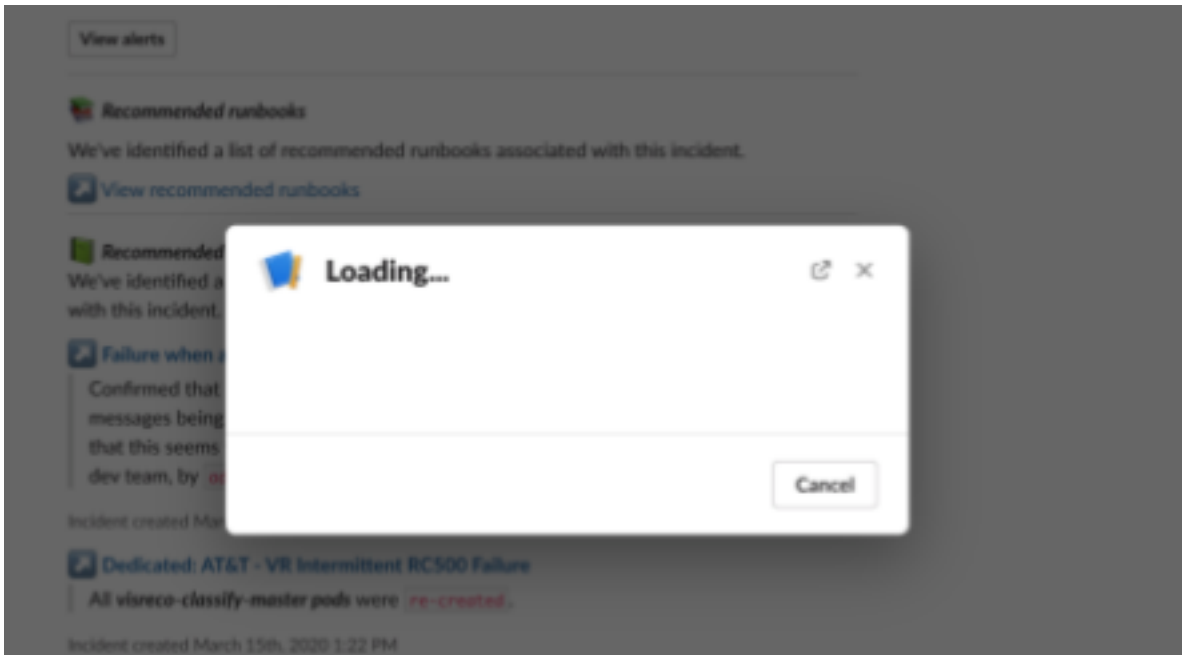


Figure. View alerts

- The **Log preview** dialog in Slack or Microsoft Teams hangs with the **`Loading log data..`** message.

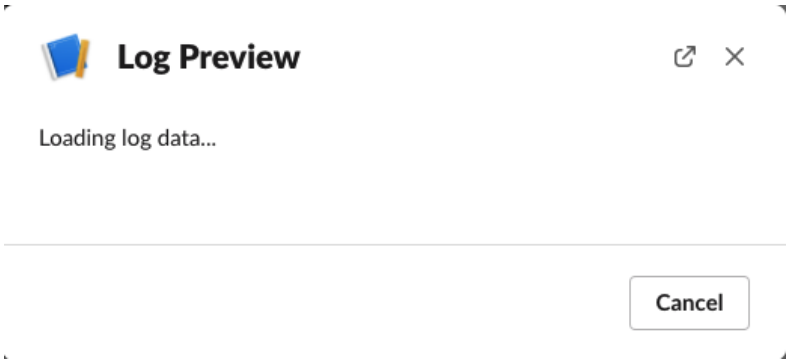


Figure. Log preview

- Checking the log results in something similar to this:

Figure. View log

Solution: close the **View alerts** dialog or the **Log preview** dialog in Slack or Microsoft Teams, and retry the operation.

Alerts and incidents not appearing in Slack

Alerts and incidents are showing up on IBM Cloud Pak Automation console but not in Slack ChatOps.

A number of causes can be considered when alerts are not showing up in Slack. Try one or more of the following to help resolve the problem.

1. Make sure that alerts and incidents are being created from events: alerts and incidents can be seen in the *Incidents and alerts* page in the IBM Cloud Pak Automation console. If events are flowing but no alerts or incidents are being created, review Troubleshooting alert management.
2. Double check channel IDs: if alerts and incidents are visible in the IBM Cloud Pak Automation console, make sure the channel IDs are correctly pasted in the ChatOps data connection form. Ensure the channel ID rather than the channel name is being used and make sure that no extra blank spaces are in the form field. Instructions for retrieving channel IDs can be found in the Configuring Slack applications for connection page.
3. Ensure that the Slack app is a channel member: to post incident messages, the IBM Cloud Pak for AIOps Slack app must be a channel member of the channel IDs being used. To add an application to a channel, tag the bot with the "@" symbol followed by the bot name.
4. Look for any visible errors in the **`chatops-slack-integrator`** logs. If the IDs are correct and the app is a channel member, look for error messages in the pod and contact the L2 support team.

Note: If the Slack app is not properly configured when an incident is initially created, that incident is not posted to Slack. Only when the app is successfully configured do new incidents, and updates to those incidents, get posted.

Slack ChatOps connection is not working within an air-gapped (offline) environment

If you create a Slack connection within an air-gapped (offline) environment, the connection might not work as expected. When you first create the Slack connection, the test of the connection (by clicking **Test connection to Slack.com.**) must be successful. If the test was successful, but the Slack connection is still not working, restart the Slack integrator pod to retrieve the latest connection information.

To restart the pod, complete the following steps:

1. From a command line, use the OpenShift CLI **`oc login`** command to log in to your cluster.
2. Switch to the project (namespace) where IBM Cloud Pak for AIOps is installed:

   ```
   oc project <project>
   ```

   Where **`<project>`** is the project (namespace) where IBM Cloud Pak for AIOps is installed.
3. Retrieve Slack integrator pod name:

   ```
   oc get pods | grep slack
   ```

   Record the name of the Slack integrator pod.
4. Delete the pod to cause the pod to restart:

   ```
   oc delete pod <slack integrator pod>
   ```

   Where **`<slack integrator pod>`** is the name of the pod name.

Microsoft Teams ChatOps connection shows no response in the console

If you have a Microsoft Teams ChatOps connection, you can encounter an issue the Microsoft Teams console shows no response through the connection. This issue can occur when you install and configure a Secure Tunnel for IBM Cloud Pak for AIOps on an on-premises Red Hat OpenShift Container Platform and configure a corresponding Secure Tunnel connecter for Microsoft Teams on a virtual machine (VM) that is accessible from a public network.

If you encounter this issue, complete the following steps:

1. Verify that the Secure Tunnel is working.
   1. Log in to the IBM Cloud Pak for AIOps console.
   2. Go to **Administration** > **Secure Tunnel**.
   3. Verify that the Secure Tunnel connection is configured and that the status is **`Ready`**.
   4. Click into the **Application mappings** column of your Secure Tunnel connection. Verify the URLs list in the **Application mapping address** column.

   Notes:

- If the status is not **ready**, then check whether the Secure Tunnel Connector is installed and running. For more information about setting up the connector, see Install the Secure Tunnel Connector.
- If your Secure Tunnel Connector is running, but you do not see the status as **Ready** for the connection, check the log messages from the Secure Tunnel Connector container.
  - If the log message indicates TLS errors, you might not have the correct Secure Tunnel Connector package. Verify that the installed package is correct. If necessary, download and install the correct package. For more information about installing the connector, see Install the Secure Tunnel Connector.
  - If the log message indicates a connecting error, check whether you can access the virtual machine (VM) where the Secure Tunnel Connector is installed from the Secure Tunnel pod.

2. Verify that the network between the Secure Tunnel and the VM on which Secure Tunnel Connector is installed is accessible.

   1. Log in to the Red Hat OpenShift Container Platform console where IBM Cloud Pak for AIOps is installed.
   2. Go to **Workloads** > **Pods**.
   3. Switch to the project (namespace) for IBM Cloud Pak for AIOps.
   4. Filter the pods to find the secure tunnel pod. For instance filter by **sre-tunnel-network**.
   5. Click listed pod. Then, click **Terminal**.
   6. Access the IP address of your VM by running a command similar to the following **curl** command on port **50443**:

      ```
      curl -k https://your.vm.external.host.com:50443
      ```

      - If you get any SSL error from the command, the error indicates that the network connection is working.
      - If you a message tht indicates **Connection refused** from the command, the message indicates that the network connection is not reachable. Contact your network administrator to fix the network connection.

3. Verify that the configuration of the Microsoft Azure Bot for the ChatOps connection is correct.

   1. Log in to the Microsoft Azure portal, and click **Bot services**. Then, click your Azure bot.
   2. Click **Configuration** under **Settings**.
   3. Check the **messaging endpoint** setting. Replace the **request URL** value with the **application mapping URL** from the Secure Tunnel connection. For more information about the settings for the connection, see Creating a Microsoft Teams connection.
   4. Attempt to send the message **@<bot_name> /waiops_welcome** in Microsoft Teams. If the connection still does not work, other issues might exist. contact your network administrator for assistance with investigating and fixing the network connection.

4. Verify that the network between the Microsoft Azure Bot and the VM on which the Secure Tunnel connector is installed is accessible:

   1. Log in to the terminal of the VM on which you installed the Secure Tunnel connector.
   2. Find the network interface (NIC) that is accessible from public network. You can use a command like **ip address** to find this interface. For example, if your network interface is **eth1**, listen to the port **12443** on that network interface by using the command **tcpdump -i eth1 port 12443**.
   3. Test the access to the **application mapping URL** of your Secure Tunnel connection from your device or from another device outside of your on-premise netowork. For example, run the following command, where **your.vm.external.host.com** is your VM host.

      ```
      curl -k -X POST
      https://your.vm.external.host.com:12443/aiops/aimanager/instances/1683613041902750/teams/api/messages
      ```

   4. Verify that there are outputs from the **tcpdump** command while you run the **curl** command.

      If you do not find any output from the **tcpdump** command, you might have firewalls or ACL restrictions that are blocking inbound traffic to the VM. Contact your network administrator to grant inbound traffic on port **12443** for the VM and retest the command.

During a ChatOps secure tunnel creation an 'installation failed' message displays

When you create a ChatOps connection and it fails, wait for a few minutes to see if the installation retries, and if it does not simply create the connection again.

Troubleshooting topology

Learn how to isolate and resolve problems with the Topology viewer and topology service.

To help you with troubleshooting topology issues, review the topology service logs. For more information, see Viewing the topology logs.

Common issues:

- Chrome extension display issues on Incident Topology page
- Topologies not visible on Incident Topology page after resource merge
- Error connecting topology services to Cassandra database
- Observer schedule toggle becomes disabled after saving the connection
- File Observer fails due to hidden characters
- Halt a Cassandra node for maintenance
- Topology observer job failure due to prohibited egress

Chrome extension display issues on Incident Topology page

When displaying the details for a listed event in the Resolution Hub by using the Chrome browser, the resizing of the page might cause screen errors. This is caused by the MEGA Secure Cloud Storage and Chat Chrome browser extension.

Solution: Remove the MEGA Secure Cloud Storage and Chat extension.

Topologies not visible on Incident Topology page after resource merge

When resources from two observer sources have been merged by using the topology merge function, the topology is no longer displayed in the Incident view. This known issue affects **only** the Incident view, and the topology is still present in all other views.

Error connecting topology services to Cassandra database

During startup, the topology service can attempt to connect to the Cassandra database before it is fully started, which causes an error, followed by a warning message similar to the following message:

```
WARN   [2022-02-28 11:01:36,748] [main] c.i.i.t.g.ConnectionManager -  failed to connect to storage back end sleeping before retrying.
```

This issue can occur when the topology service has started before the Cassandra database is fully ready.

Solution: You do not need to complete any workaround. The topology service tries to connect to the Cassandra database repeatedly. When the Cassandra database is up and running, the service can successfully connect.

Observer schedule toggle becomes disabled after saving the connection

After editing and saving an observer connection for an observer with scheduled jobs, the **Schedule request** toggle becomes disabled. However, the job remains scheduled and continues to run in the background.

This issue can occur when the **Schedule request** toggle fails to read the existing job schedule and erroneously reverts to its default setting of **Off**.

Solution: The job scheduling remains active and no workaround is required. To confirm the existing observer job schedule:

1. Log in to IBM Cloud Pak Automation console.
2. Expand the navigation menu (four horizontal bars), then click **Define** > **Data and tool connections**.
3. On the *Data and tool connections* page, click **Add connection**.
4. On the *Add connections** page, click **Topology** in the *Category* list that is next to the list of all connections.
5. Click the **configure, schedule, and manage other observer jobs** link in the description for the topology connections.
6. Select the observer.
7. Confirm that the job schedule remains active.

File Observer fails due to hidden characters

An error can occur when your File Observer input file contains hidden characters that interfere with the processing of the content.

```
The following lines numbers had problems, check the logs for details: 1
```

This issue can occur when a file appears compliant in content and format, but is in a UTF-8 Unicode (with BOM) file format (instead of the regular UTF-8 file format).

Solution: Change the file format. For example, you can create a new file from the source file by using the following command:

```
sed '1s/^\xEF\xBB\xBF//' < topology.txt > new.txt
```

Halt a Cassandra node for maintenance

Sometimes you need to manually maintain the Cassandra server nodes. For example, you might need to halt one node for long enough to allow you to complete the required maintenance work or debug a problem.

The following steps suspend (halt) one Cassandra node while the other nodes still serve queries, which allows you to perform maintenance tasks without shutting down your production environment.

Note: To reduce the impact of pausing a Cassandra node on the performance of your remaining system resources, keep such maintenance periods to a minimum.

Halt the node

1. Edit the bootstrap configMap.

```
oc edit configmaps aiops-topology-cassandra-bootstrap-config
```

For example, if you want to halt node 0, you can set the following columns.

```
hostname_filter: aiops-topology-cassandra-0
running_mode: maintenance
```

2. Save and exit.

3. To restart the target node pod, you delete the pod so that the statefulSet will restart the pod automatically. Then it should be halted.

```
oc delete pod aiops-topology-cassandra-0
```

Debug the Cassandra server

After the node is restarted and halted, you should see that its state stays as **`0/1 Running`**. When the pod is in this mode, the Cassandra server is not automatically started in it. The **`liveness`** probe and **`readiness`** probe will not be activated for many days in this pod, allowing you to perform any maintenance that is required.

There are two methods to start the Cassandra server for debug purposes.

- You can go into the pod container to manually start the server.

  ```
  oc exec -it aiops-topology-cassandra-0 -- bash
  bash-4.4$ /opt/ibm/cassandra/bin/cassandra -fR > /tmp/server.log 2>&1 &
  ```

  The server process will be started in the background and the log messages can be found in file **`/tmp/server.log`**.

- You can reuse the entry point script.

  ```
  oc exec -it aiops-topology-cassandra-0 -- bash
  bash-4.4$ export RUNNING_MODE=normal
  bash-4.4$ /opt/ibm/start-cassandra.sh > /tmp/server.log 2>&1 &
  ```

Example maintenance task: Remove corrupt commit logs that are causing Cassandra to not start.

```
$ oc exec -it aiops-topology-cassandra-0 -- bash
bash-4.4$ rm -rf /opt/ibm/cassandra/data/commitlog/CommitLog-6-1682019553495.log
```

After performing the required maintenance, you can stop the manually started Cassandra server.

```
oc exec -it aiops-topology-cassandra-0 -- bash
bash-4.4$ nodetool stopdaemon
```

Note: You can also perform other actions, like fixing volumes or transforming the data files. Every time you restart the pod, it will be halted.

Return the node to normal

After you complete your manual work, you can edit the configMap and restart the pod to return it back to normal.

```
oc edit configmaps aiops-topology-cassandra-bootstrap-config
```

Set the running mode to normal.

```
hostname_filter: aiops-topology-cassandra-0
running_mode: normal
```

Restart normal operations

Finally, you can delete the pod again so it will revert to the normal node and continue to work in the cluster.

Topology observer job failure

When you attempt to run a topology observer, the job fails because no response is returned for the request to the corresponding external service.

Solution: Run the following commands to create a NetworkPolicy that allows egress from the topology observer pods:

```
export AIOPS_NAMESPACE=<AIOps installation namespace>

cat << EOF | oc apply -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: aiops-observer-egress
  namespace: ${AIOPS_NAMESPACE}
spec:
  egress:
  - {}
  podSelector:
    matchLabels:
      app.kubernetes.io/instance: aiops-topology
  policyTypes:
  - Egress
EOF
```

Viewing the topology logs

You can see all topology and observer logs by using the **oc logs** command.

About this task

All observers are deployed in pods. You can identify logs either by their names or their labels.

To increase or decrease the log levels for a specific observer to help investigate issues, you first use the **oc get deployments** command to display a list of all deployments. Then, use the **oc edit deployment** command to change the logging levels to, in this example, 'debug'. A log level of 'debug' logs all request or response operations to provide detailed diagnostic data.

Viewing logs

1. To list all pods and the labels that are applied to the pods, run the **oc get pod --show-labels** command.

   ```
   oc get pod --show-labels
   ```

   Sample output:

   ```
   deploy-test-ciscoaci-observer-8755fcc94-7cp7h      1/1       Running   0          2d
   app=ciscoaci-observer,chart=ciscoaci-observer,heritage=Tiller,pod-template-hash=431197750,release=deploy
   deploy-test-file-observer-76c5869d8b-tm7xj         1/1       Running   0          2d        app=file-
   observer,chart=file-observer,heritage=Tiller,pod-template-hash=3271425846,release=deploy-test
   deploy-test-rest-observer-789cd8699d-zb8vx         1/1       Running   0          2d        app=rest-
   observer,chart=rest-observer,heritage=Tiller,pod-template-hash=3457842558,release=deploy-test
   deploy-test-vmvcenter-observer-675bd88f5c-8kgnk    1/1       Running   0          2d
   app=vmvcenter-observer,chart=vmvcenter-observer,heritage=Tiller,pod-template-
   hash=2316844917,release=deploy-test
   ```

2. Display the logs by using either the pod names or labels. When you run the **oc** logs command, you must also specify the project (namespace).

   1. View logs for a specific pod:

      ```
      oc logs asm-layout-88bd88bdb-htjfp --namespace=netcool
      ```

   2. View logs by using the label:

      ```
      oc logs -l app=layout --namespace=netcool
      ```

Changing log levels to debug

Run the following command to change the debug level to **DEBUG**:

```
oc exec -it deploy-test-file-observer-76c5869d8b -- /opt/ibm/netcool/asm/bin/log-level.pyc --level DEBUG
```

Sample system output

```
Configured logging level for com.ibm to DEBUG
Response code: 200
```

Troubleshooting access control and user management (IAM)

Troubleshoot console access control and Identity and Access Management (IAM) user management issues.

- Login issue
- LDAP issue
- Pod issues

Login

Review frequently encountered login issues for the IBM Cloud Pak Automation console.

- Cannot log in to the Automation console by using the admin username
- Cannot log in to the Automation console with OpenShift authentication
- Cannot log in to the Automation console after reinstallation of foundational services
- Default admin cannot log in due to LDAP server issues
- Login too slow or times out, or invalid username or password error
- Cloudctl login fails
- Docker login results in unencrypted password warning

Troubleshooting Infrastructure Automation install and uninstall

Review frequently encountered issues related to installing and uninstalling Infrastructure Automation.

- Troubleshooting installing and uninstalling Infrastructure Automation
- Troubleshooting Infrastructure Management
- Troubleshooting Managed services

Troubleshooting installing and uninstalling Infrastructure Automation

Review frequently encountered issues related to installing and uninstalling Infrastructure Automation.

- Uninstall fails to remove Infrastructure management cleanly after deleting IAConfig instance
- Database is stuck on retrying on Infrastructure Automation IMInstall Power Environment
- The **`cam-tenant-api`** pod is not in a ready state after installing the IAConfig CR
- After uninstalling IBM Cloud Pak for AIOps, headings are missing in the Infrastructure Automation UI, and unable to log into Infrastructure management UI
- Offline install or upgrade stuck because the 'oc ibm-pak generate mirror-manifests' command fails with 'no space left on device'

Uninstall fails to remove Infrastructure management cleanly after deleting IAConfig instance

After deleting Infrastructure management IAConfig CR in the namespace to uninstall Infrastructure management from Red Hat OpenShift Container Platform console, some of the Infrastructure management pods were still there in the namespace. You can workaround the problem by deleting IMInstall custom resource from the Red Hat OpenShift Container Platform console. This will remove the pods from the namespace.

You should also delete the **`clients.oidc.security.ibm.com`** customer resource that was used by Infrastructure management if it still exists. Then, You can uninstall the operators remaining on the namespace.

To delete IMInstall Custom Resource (CR), you need to edit the CR to remove the Finalizer. Save the CR. Then delete the IMInstall Custom Resource (CR). Run the following commands or perform the same from the Red Hat OpenShift Container Platform console.

1. Identify the CR name, for example, **`im-iminstall`**

   ```
   oc -n <project> get iminstall
   ```

   Where **`<project>`** is the project (namespace) where Infrastructure Automation is deployed.

2. Edit the statement "finalizers:" and the one below it. Save and exit.

```
oc -n <project> edit iminstall im-iminstall
```

Where **<project>** is the project (namespace) where Infrastructure Automation is deployed

3. Delete the CR.

```
oc -n <project> delete iminstall im-iminstall
```

Where **<project>** is the project (namespace) where Infrastructure Automation is deployed

After a few minutes, the Infrastructure management pods should no longer exist. The clients.oidc.security.ibm.com Custom Resource (CR) could still exist, and to delete the CR, do the following steps,

1. Check if the CR exist.

```
oc -n <project> get clients.oidc.security.ibm.com
```

Where **<project>** is the project (namespace) where Infrastructure Automation is deployed

2. If the CR exists, edit the CR. Delete the statement "finalizers:" and the one below. Save and exit.

```
oc -n <project> edit clients.oidc.security.ibm.com ibm-infra-management-application-client
```

Where **<project>** is the project (namespace) where Infrastructure Automation is deployed

3. Delete the CR.

```
oc -n <project> delete clients.oidc.security.ibm.com ibm-infra-management-application-client
```

Where **<project>** is the project (namespace) where Infrastructure Automation is deployed

Database is stuck on retrying on Infrastructure management IMInstall Power Environment

Sometimes when Infrastructure management is deployed, the orchestrator pod fails to start properly with the following in the oc logs:

```
Cannot connect to the database!
Deployment status is
```

Check the postgresql pod to ensure it's running and listening. Then, verify the **vmdb_production** database was created by rushing into the postresql pod:

```
sh-4.4$ psql -U postgres
psql (10.19)
Type "help" for help.

postgres=# \l
                                  List of databases
      Name       |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
-----------------+----------+----------+------------+------------+
 postgres        | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0       | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
                 |          |          |            |            | postgres=CTc/postgres
 template1       | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
                 |          |          |            |            | postgres=CTc/postgres
(3 rows)
```

Solution: If the **vmdb_production** database doesn't exist, like above, it is possible that a remnant postgresql.conf in the DATA directory is preventing the postgresql image from initializing the new database on boot.

You will need to check the DATA directory and either clean it manually or recreate the PVC/PV and check that it is empty.

You can always find the location of the DATA directory using the psql command below:

```
postgres=# show data_directory;
        data_directory
------------------------------
 /var/lib/pgsql/data/userdata
(1 row)
```

Ensure that the data directory is empty before the postgresql pod is restarted.

The `cam-tenant-api` pod is not in a ready state after installing the IAConfig CR

After you install Infrastructure Automation, you can encounter an error where the **cam-tenant-api** pod displays as running, but not in a ready state. When this error occurs, you can see the following message:

```
[ERROR] init-platform-security - >>>>>>>>>> Failed to configure Platform Security. Will retry in 60 seconds
<<<<<<<<<<< OperationalError: [object Object]
```

If this error occurs, delete the **cam-tenant-api** pod to cause the pod to restart and attempt to enter a ready state.

After uninstalling IBM Cloud Pak for AIOps, headings are missing in the Infrastructure Automation UI, and unable to log into Infrastructure management UI

If IBM Cloud Pak for AIOps and Infrastructure Automation are both installed, removing IBM Cloud Pak for AIOps can cause a translation file that is still in use by Infrastructure Automation and Infrastructure management to be deleted.

Solution:

1. Create a copy of the **post-camroles-translations-job**.

   ```
   oc get job post-camroles-translations-job -n <project> -o yaml > rerun-post-camroles.yaml
   ```

   Where **<project>** is the project (namespace) where Infrastructure Automation is deployed.

2. Update the **rerun-post-camroles.yaml** file to remove the job information that is specific to the previous job.

   1. Change name to **rerun-camroles-translation-jobs**.
   2. Remove **metadata.ownerReferences section**.
   3. Remove **metadata.resourceVersion**.
   4. Remove **metadata.uid**.
   5. Remove **spec.selector** section.
   6. Remove **spec.template.metadata** section.

3. Apply the updated YAML file.

   ```
   oc apply -f rerun-post-camroles.yaml
   ```

Offline install or upgrade stuck because the 'oc ibm-pak generate mirror-manifests' command fails with 'no space left on device'

The **oc ibm-pak generate mirror-manifests $IA_CASE_NAME $TARGET_REGISTRY --version $IA_CASE_VERSION** command fails with a message similar to the following in **$IBMPAK_HOME/.ibm-pak/logs/oc-ibm_pak.log**:

```
write /tmp/render-unpack-4002583241/var/lib/rpm/Packages: no space left on device
```

Solution: The default temporary directory does not have enough space to run the **ibm-pak** tool. You must set the **TMPDIR** environment variable to a different directory with more space before running the **oc ibm-pak generate mirror-manifests** command.

```
TMPDIR=<new_temp_dir> oc ibm-pak generate mirror-manifests $IA_CASE_NAME $TARGET_REGISTRY --version
$IA_CASE_VERSION
```

Where **<new_temp_dir>** is the path for a directory with more space.

Troubleshooting for Infrastructure management

Learn how to isolate and resolve problems with Infrastructure management.

To get support, see Support in the product documentation.

- After upgrading to IBM Cloud Pak for AIOps users cannot access Infrastructure management
- Error when using multiple tabs in the same browser
- Non-LDAP users cannot access Infrastructure management
- The vmware-infra-* pods in CrashLoopBackOff error states
- Infrastructure management vm operator encounters an error creating CRD
- Infrastructure management vm operator encounters an error when upgrading on OpenShift
- Error when deleting cloud volumes in Infrastructure management
- Error when logging out of Infrastructure management
- No new data coming in to IBM Cloud Pak® for AIOps from Infrastructure management

After upgrading users cannot access Infrastructure management

After upgrading, an OIDC provider error can be seen when you access Infrastructure management.

This error can result in the following error message:

```
OpenID Connect Provider error: Error in handling response type.
```

Solution: To resolve this issue, complete the following steps:

1. Log in to your Red Hat OpenShift cluster:

   ```
   oc login
   ```

2. Change to the Infrastructure management project:

   ```
   oc project management-infrastructure-management
   ```

3. Save the secret encoded string to a file and use the **base64** command to decode the data and direct the output to another text file.

   ```
   oc get secret -o jsonpath='{.data.oidc\.conf}' ibm-infra-management-application-connection | base64 -d |
   sed -e 's#oidc/endpoint/OP/jwk#idprovider/v1/auth/jwk#g'> oidc_new.conf
   ```

   ```
   cat oidc_new.conf | base64 | tr -d '\n'
   ```

4. Replace the value above in the secret **ibm-infra-management-application-connection** in the **oidc.conf**

   ```
   oc edit secret ibm-infra-management-application-connection
   ```

   - Replace oidc.conf with the new value in step 3.
   - Save the secret.

5. Restart the **httpd** pod in the **management-infrastructure-management** namespace to pick up the changes by deleting the **<httpd_pod_instance_name>**. For example,

   ```
   oc delete pod `oc get pod | grep httpd | cut -f1 -d" "`
   ```

Error with multiple tabs in the same browser by using Infrastructure management

When you are using Infrastructure management, you can encounter an error starting multiple tabs in the same browser.

You can encounter the following error when you attempt to access Infrastructure management using multiple tabs in the same browser.

```
Edit aborted! Infrastructure management does not support the browser's back button or access from multiple
tabs or windows of the same browser. Close any duplicate sessions before proceeding.
```

This error occurs when Infrastructure management does not support the browser's back button or access from multiple tabs or windows of the same browser.

Solution: Closing the other Infrastructure management browser tab can resolve the issue.

Non-LDAP users cannot access Infrastructure management but can see the data in IBM Cloud Pak for AIOps

While you are logged in to IBM Cloud Pak® for AIOps with a non-LDAP user, attempting to start Infrastructure management can fail with an error.

When this error occurs, an error message can resemble the following example:

```
OpenID Connect Provider error: Error in handling response type.
```

When you attempt to access the Infrastructure management. console through the side panel, you are sent to **inframgmtinstall.apps.<cluster_name>.<customer_domain>.com** as expected. This link brings you to a login page.

If you log in with an LDAP user, the **initialAdminGroupName** must exist as an LDAP group. Non-LDAP users cannot be used with Single Sign On for Infrastructure management.

Solution: The initialAdminGroupName that was provided in the IMConfig was not a valid group in the enterprise LDAP directory, instead it was a group manually setup in the Identity and Access Management (IAM) panel. The user that attempted to access Infrastructure management. using this group was manually added to the group, giving the illusion of a successful integration with the LDAP directory.

The initialAdminGroupName must be a valid group within the enterprise LDAP directory and the designated user must be added to this group. Once the user logs into the IBM Cloud Pak for AIOps console, they will be automatically added to the Identity and Access Management panel.

The vmware-infra-* pods in CrashLoopBackOff error states

You can encounter an error where several pods are crashing, and restarting constantly. These pods can appear to be failing with an authentication error.

When this error occurs, your output when checking the pod status can resemble the following example:

```
NAMESPACE                                                 NAME
READY   STATUS              RESTARTS   AGE
management-infrastructure-management                      1-vmware-infra-event-catcher-30-65d7bfd967-tcsrg
0/1     CrashLoopBackOff    399        44h
management-infrastructure-management                      1-vmware-infra-operations-30-6dc9f87b75-mqfpz
0/1     CrashLoopBackOff    402        44h
management-infrastructure-management                      1-vmware-infra-refresh-30-84fcd6cbd4-fc68w
0/1     Error               412        44h
```

When you check the logs, these pods appear to be failing with an authentication error:

```
{"@timestamp":"2020-08-02T21:13:00.050303 ","hostname":"1-vmware-infra-refresh-30-84fcd6cbd4-
fc68w","pid":7,"tid":"2ae9ee5f997c","level":"err","message":"MIQ(ManageIQ::Providers::Vmware::InfraManager::R
efreshWorker::Runner) ID [2241] PID [7] GUID [403658a2-96a2-4823-b996-94cd6472dffd] EMS id [30] failed
authentication check. Worker exiting."}
```

This error can happen when VMWare is unable to be accessed, which can be due to an authentication issue, network issue, or some other reason. When this connection issue occurs, the pods enter into this state.

Solution: To resolve this error, complete the following steps:

1. Fix the credentials or network issue, allowing the pods to recover.
2. Delete the provider to stop the CrashLoopBackOff.

Infrastructure management vm operator encounters an error when creating CRD

After you install Infrastructure management, you can encounter an error where no **deployers** are in the **management-infrastructure-management** namespace.

```
oc get deployers.core.hybridapp.io --all-namespaces
```

Output:

```
No resources found
```

This error can occur when the operator attempts to create the CR before the CRD is installed. To check whether this issue is occurring, run the following command:

```
oc logs -n management-infrastructure-management -f infra-management-vm-operator-764ff5fff9-zm8lf
```

Output:

```
I0729 23:17:27.105548       1 deployer.go:81] Create Deployer: error creating CRD, the server could not find
the requested resource
```

Solution: Restart the **infra-management-vm-operator** pod. When the pod is restarted run the following command to view the result:

```
oc get deployers.core.hybridapp.io --all-namespaces
```

Output:

```
NAMESPACE                              NAME        AGE
management-infrastructure-management   ibminfra    12s
```

Infrastructure management vm operator encounters an error when upgrading on OpenShift

When you are upgrading Infrastructure management, you can encounter an error where the operator fails to install correctly.

Checking the OpenShift console InstalledOperators for project: **management-infrastructure-management** shows the operator in "Failed" state. Checking **oc describe** shows a message "conflicting CRD owner in namespace". For example,

```
oc describe operator ibm-management-infra-vm.management-infrastructure-management
```

```
Name:         ibm-management-infra-vm.management-infrastructure-management
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  operators.coreos.com/v1
Kind:         Operator
Metadata:
  ...
  ...
      Namespace:                management-infrastructure-management
      API Version:              operators.coreos.com/v1alpha1
      Conditions:
        Last Transition Time:  2020-11-23T23:19:46Z
        Last Update Time:      2020-11-23T23:19:46Z
        Message:               installing: missing deployment with name=infra-management-vm-operator
        Reason:                NeedsReinstall
        Status:                True
        Type:                  Pending
      Kind:                    ClusterServiceVersion
      Name:                    ibm-management-infra-vm.v2.2.3-202011230408
      Namespace:               management-infrastructure-management
      API Version:             operators.coreos.com/v1alpha1
      Conditions:
        Last Transition Time:  2020-11-23T23:19:44Z
        Last Update Time:      2020-11-23T23:19:44Z
        Message:               conflicting CRD owner in namespace
        Reason:                OwnerConflict
        Status:                True
        Type:                  Failed
      Kind:                    ClusterServiceVersion
      Name:                    ibm-management-infra-vm.v2.2.3-202011231858
      Namespace:               management-infrastructure-management
Events:                        <none>
```

This error CAN be seen when conflicting CSVs exist:

```
oc get csv -n management-infrastructure-management
```

Output:

```
NAME                                                  DISPLAY
VERSION              REPLACES                                                  PHASE
ibm-management-infra-vm.v2.2.3-202011231109           IBM Management Infrastructure for virtual machines
2.2.3-202011231109   ibm-management-infra-vm.v2.2.3-202011230859          Replacing
ibm-management-infra-vm.v2.2.5-202012021409           IBM Management Infrastructure for virtual machines
2.2.5-202012021409   ibm-management-infra-vm.v2.2.3-202011231109          Succeeded
```

Solution: Delete any older CSVs and this will allow the operator to install.

```
oc delete csv -n management-infrastructure-management ibm-management-infra-vm.v2.2.3-202011231109
```

```
oc delete csv -n management-infrastructure-management ibm-management-infra-vm.v2.2.3-202011230859
```

Error when deleting cloud volumes in Infrastructure management

After you create a cloud volume, you might be unable to detach the volume from an instance. When this issue occurs, you can view an error if you then try to delete the volume. The error can resemble the following message:

```
Unable to detach Cloud Volume "volume_name" from "instance_name" Instance "instance_ID" is not attached to
volume "volume_ID".
```

This error can occur when the refresh (EmsRefresh) has not run or completed. to resolve this issue, manually run a refresh by browsing to **Configuration** > **Refresh Relationships and Power States**. Then, wait for the refresh to complete. When the refresh is completed, delete or detach the volume again.

Errors logging out of or re-logging in to Infrastructure management

While logged in to IBM Cloud Pak® for AIOps and the IBM Cloud Pak Automation console you log out of Infrastructure management instance and you receive an error. The error states:

```
This page isn't working right now.  HTTP ERROR 400
```

Logging out of Infrastructure management takes you back to the login for IBM Cloud Pak for AIOps. When attempting to start the IBM Cloud Pak Automation console it fails to log you back in through SSO and you are prompted to enter login credentials.

You can log out of IBM Cloud Pak for AIOps, then log back in to IBM Cloud Pak for AIOps and start the IBM Cloud Pak Automation console.

No new data coming in to IBM Cloud Pak® for AIOps from Infrastructure management

You can encounter an issue where providers are added within Infrastructure management, but are not showing IBM Cloud Pak for AIOps.

This issue can occur when the access token that provides authorization into Infrastructure management from IBM Cloud Pak for AIOps is expired.

To confirm this cause, check the BypassAuth annotation in the **imconnection connections.infra.management.ibm.com** CR in the **management-infrastructure-management** project or namespace.

```
metadata:
  annotations:
    BypassAuth: 'false'
```

If the BypassAuth annotation is not defined or is set to "false" then:

- check if spec.cfHost is defined with the correct Infrastructure management url.
- check for the definition of spec.secrets.accessToken.

```
spec:
  cfHost: 'inframgmtinstall.apps.cp4mcm-23.company.com'
  secrets:
    accessToken:
      secretKeyRef:
        key: accesstoken
        name: ibmconnectionsecret
```

If accessToken is defined then check the log of the VM operator and look for entries with "TokenRefresh Token has expired" similar to the following:

```
I0401 13:59:35.553292       1 update.go:34] TokenRefresh Starting update of token.
I0401 13:59:35.555424       1 secret.go:166] TokenRefresh Getting secrets from management-infrastructure-
management using name imconnectionsecret
I0401 13:59:35.571513       1 secret.go:173] TokenRefresh got token secret 131462371
I0401 13:59:35.571540       1 update.go:94] TokenRefresh checking if expired.
W0401 13:59:35.571553       1 secret.go:266] TokenRefresh Token has expired. Current time: 2021-04-01
13:59:35.571549714 +0000 UTC m=+29212.072337265, End time: 2021-03-26 18:52:49 +0000 UTC
E0401 13:59:35.571570       1 update.go:65] refresh token has elapsed not attempting refresh
E0401 13:59:35.571615       1 update.go:27] error getting token using refresh token
I0401 13:59:35.571622       1 configs.go:92] Sleeping for 300.000000 seconds.
```

Solution

If the accessToken has expired then update the secret referenced in **accessToken.secretKeyRef.name**.

Note: For version 2.3 upgraded from 2.2, the default secret name is **imconnectionsecret**. For a new install of version 2.3 the default secret name is **ibm-infra-management-application-connection**.

To fix the problem the oauth username and password can be added to the **<secret_name>** in order for the tokenrefresh process to request a new accesstoken from IAM. Once the accesstoken is obtained the oauth username and password are removed from the secret.

Edit the secret using the command:

```
oc edit secret <secret_name> -n management-infrastructure-management -o YAML
```

Add after the line **type: Opaque**

```
type: Opaque
stringData:
  oauth_username: <user_name>
  oauth_password: <password>
```

Where **<secret_name>** is **ibm-infra-management-application-connection**

If the BypassAuth annotation is set to "true" then check to make sure spec.cfHost in the **imconnection** CR is set to the following:

```
spec:
  cfHost: 'web-service.management-infrastructure-management.svc.cluster.local:3000'
```

Verifying the solution

Check the log for the **sre-inventory-inventory-cfcollector** pod in the **kube-system** project for entries similar to the following:

```
I0425 00:13:35.835415       7 main.go:41] ============Do synchronization from CloudForm http://web-
service.management-infrastructure-management.svc.cluster.local:3000
I0425 00:13:39.922796       7 query_vms.go:147] Totally 333 vms found in cloudform. 333 VMs should be
returned and actually 333 returned in this query.
I0425 00:13:40.029993       7 query_providers.go:78] Totally 4 providers found in cloudform. 4 providers
should be returned and actually 4 returned in this query.
I0425 00:13:40.114010       7 query_availabilityzone.go:77] Totally 0 availability_zones found in cloudform.
0 availability_zones should be returned and actually 0 returned in this query.
I0425 00:13:40.309270       7 query_hosts.go:112] Totally 31 hosts found in cloudform. 31 hosts should be
returned and actually 31 returned in this query.
I0425 00:13:40.387620       7 query_services.go:84] Totally 0 services found in cloudform. 0 services should
be returned and actually 0 returned in this query.
I0425 00:13:40.470200       7 query_cloud_volumes.go:104] Totally 0 cloud volumes found in cloudform. 0 cloud
volumes should be returned and actually 0 returned in this query.
I0425 00:13:40.627044       7 query_data_stores.go:105] Totally 43 data stores found in cloudform. 43 data
stores should be returned and actually 43 returned in this query.
I0425 00:13:41.366165       7 query_cloud_networks.go:101] Totally 112 cloud networks found in cloudform. 112
cloud networks should be returned and actually 112 returned in this query.
I0425 00:13:42.101278       7 query_cloud_subnets.go:97] Totally 342 cloud subnets found in cloudform. 342
cloud subnets should be returned and actually 342 returned in this query.
I0425 00:13:42.176657       7 query_network_routers.go:92] Totally 0 network router found in cloudform. 0
network routers should be returned and actually 0 returned in this query.
I0425 00:13:42.451049       7 query_security_groups.go:95] Totally 112 security groups found in cloudform.
112 security groups should be returned and actually 112 returned in this query.
I0425 00:13:42.959106       7 query_floating_ips.go:96] Totally 287 floating ips found in cloudform. 287
floating ips should be returned and actually 287 returned in this query.
I0425 00:13:42.961653       7 sender.go:206] Sending Resources { request: 99999, add: 853, update: 147,
delete: 0, edgedd: 0, edgeDelete:0}
I0425 00:13:43.427390       7 sender.go:206] Sending Resources { request: 99999, add: 0, update: 200, delete:
0, edgedd: 800, edgeDelete:0}
I0425 00:13:46.473678       7 sender.go:206] Sending Resources { request: 99999, add: 0, update: 0, delete:
0, edgedd: 53, edgeDelete:0}
I0425 00:13:46.683429       7 main.go:43] ============Done synchronization this time
```

Troubleshooting and support

Troubleshooting known issues and support information for Managed services.

- Getting help and support
- Troubleshooting

Getting help and support

- If you need help using Managed services, visit the support page. Additionally, you can see if the problem is documented in corresponding Troubleshooting topics.
- For Managed services blogs and tutorials, see IBM Managed services Content Development Community.
- For information about opening an IBM support ticket or about support levels and ticket severities, see Contacting support and Ask the community.
- Learn more about support from the following topics:
  - Open source support in Managed services
  - Collecting data for Managed services

Troubleshooting

- Open-source support in Managed services
- Importing custom CA Certificate into Terraform provider pod
- Troubleshooting issues in installation
- Troubleshooting issues in CLI tool
- Troubleshooting issues in installation of Managed services on Red Hat OpenShift Kubernetes Service
- Troubleshooting issues in Middleware Content
- Troubleshooting deployment issues

- Troubleshooting security related issues
- Troubleshooting issues in user interface
- Troubleshooting issues in Service Composer
- Troubleshooting issues in Service library
- Troubleshooting general issues
- Troubleshooting content related issues and useful links
- Troubleshooting Terrafom related issues
- Collecting data for Managed services
- Refreshing internal certificates
- Updating the resource limits for Managed services containers

Troubleshooting general issues

Some generally known issues with Managed services are documented.

- Sometimes, Managed services may fail to retrieve the power state of a resource from the remote provider. The cause of the error may be due to the missing underlying cloud resource. As a resolution, check the cloud provider console to ensure that the underlying resource is still present. If it has been deleted from outside of Managed services, then this error message may occur.
- Managed services might face issues whenever a microservice becomes inactive. To find and restart all the inactive microservices, run the following command from the directory where you extracted the content of the Managed services compressed file on the master node:

  ```
  bash MS-remedies.sh
  ```

  Note: The script does not restart base microservices like MongoDB, Auth, and Identity.
- For information about collecting Managed services Logs, see Collecting logs using camCollector script.

Troubleshooting issues in installation

Known issues in the installation of Managed services.

- If you face issues while accessing the Managed services user interface post the installation, check your firewall settings and turn it off.
- If the Managed services entry is missing from the navigation menu **Automate infrastructure**, this might be caused by a problem during Managed services installation.

  To resolve this issue:

  - Delete the Managed services installation instance and install it again.
  - After successful installation and creation of the Managed services entry in the navigation menu **Automate infrastructure**, delete the Service library pods to restart them.
- There will be instances where the **cam-tenant-api pod** does not start even after 45 minutes and an error 'Failed to get IAM access token' is displayed. Restart the pod to resolve the issue.
- Sometimes, even after the successful deployment of Managed services, the "cam-mongo" microservice might go down unexpectedly.

  Run the following command to check the pod log:

  ```
  kubectl describe pods -n cp4aiops
  ```

  If this command does not provide you the necessary details to understand the issue, run the kubectl command to get logs from previously running container. For example, this **kubectl -n cp4aiops logs cam-mongo-5c89fcccbd-r2hv4** command results in the following output:

  ```
  exception in initAndListen: 98 Unable to lock file: /data/db/mongod.lock Resource temporarily unavailable. Is a mongod  instance already running?, terminatingConclusion: While starting the container inside "cam-mongo" pod it was unable to use the existing /data/db/mongod.lock file and hence your pod will be not up and running and you cannot acces CAM URL.Solution:
  ```

  To resolve the issue, do the following steps:

  1. Use the following pod creation yaml to spin up a container and mount the cam-mongo volume within it. It mounts the concerned pv's, that is, **cam-mongo-pv** where /data/db/ is present.

     ```
     apiVersion: v1
     kind: Pod
     metadata:
     ```

```
  name: mongo-troubleshoot-pod
spec:
  volumes:
    - name: cam-mongo-pv
      persistentVolumeClaim:
      claimName: cam-mongo-pv
  containers:
    - name: mongo-troubleshoot
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/data/db"
          name: cam-mongo-pv
```

2. Use **`docker exec -it /bin/bash`** to keep stdin open and to allocate a terminal. Run the following commands:

   ```
   cd /data/db
   rm mongod.lock
   rm WiredTiger.lock
   ```

3. Kill the pod that you created for troubleshooting.

4. Run the following command to kill the corrupted cam-mongo pod:

   ```
   kubectl delete pods -n cp4aiops
   ```

- Managed services Container Debugging (kubectl)

  When a container is not in running state, run the following kubect1 commands to describe pods and look for errors:

  ```
  kubectl -n cp4aiops get pod
  kubectl -n cp4aiops describe pod <podname>
  kubectl -n cp4aiops get pv
  kubectl -n cp4aiops describe pv <pvname>
  ```

  Look for events or error messages when describing the pods or persistent volumes that are not in health states. For example, **`CrashLoopBackoff, Pending (for a while), Init (for a while)`**.

- Run the following commands to ensure PVs are created successfully:

  ```
  kubectl -n cp4aiops describe pv cam-mongo-pv
  ```

  If PVs are not set up, follow PV setup steps before you install Managed services:

  Note: PVs must be deleted and re-created everytime Managed services is installed.

- Managed services installation fails due to an incorrect **`Worker node architecture`** value.

  The installation fails with the following error message:

  ```
  Events:
  Type     Reason        Age                            From               Message
   ----     ------         ----       ---       -------
  Warning  FailedScheduling 71s (x2 over 71s)  default-scheduler  0/1 nodes are available: 1 node(s)
  didn't match node selector.
  ```

  To resolve this issue, review the input parameter for the 'Worker node architecture' and check if the supported architecture is selected/entered correctly: **`amd64`**

- Managed services reinstall fails due to an existing **`cam-api-secret-gen-job`** left from prior Managed services installations and the job hangs indefinitely with the following error:

  ```
  Internal service error : rpc error: code = Unknown desc = jobs.batch "cam-api-secret-gen-job" already
  exists
  ```

  ```
  root@csz25087:~# kubectl -n cp4aiops get pods
  NAME                            READY   STATUS      RESTARTS   AGE
  cam-api-secret-gen-job-n5d87   0/1     Completed   0          24m
  ```

  To resolve this issue:

  1. Run the following command:

     ```
     kubectl -n cp4aiops delete job cam-api-secret-gen-job
     ```

  2. Install Managed services.

- Managed services installation fails due to an existing **template-crd-gen-job**. In the prior installations, you might see the job hang for ten minutes and then time out.

```
Internal service error : rpc error: code = Unknown desc = jobs.batch "template-crd-gen-job" already
exists
```

```
root@csz25087:~# kubectl -n cp4aiops get pods
NAME                          READY   STATUS             RESTARTS   AGE
template-crd-gen-job-wm7mj    0/1     ImagePullBackOff   0          8m27s
```

  To resolve this issue:

  1. Run the following command:

```
kubectl -n cp4aiops delete job template-crd-gen-job
```

  2. Install Managed services.

- Encountered an error **3.6.0.0 (20220113_2156) x86_64** while accessing the library page after you uninstall and reinstall Managed services.

  To resolve this issue, increase the **icpdata_addon_version** to a higher value in the configmap **cam-proxy-zen-extension**.

```
oc -n cp4aiops edit configmap cam-proxy-zen-extension
```

Troubleshooting issues in installation of Managed services on Red Hat® OpenShift® on IBM Cloud® service

- After installing the Managed services, few pods get into an error state due to permission issue caused by using the built-in storage of IBM Cloud.

The **cam-provider-terraform-api**, **cam-proxy**, and **cam-mongo** pods get into an error state.

As a resolution, do the following steps to install Managed services successfully:

1. Update the Managed services configuration parameters for **ibm-management-cam-install** and make sure to change the custom resource **roks** to **true**.

```
spec:
  manageservice:
    license:
      accept: true
    arch: amd64
    roks: true
```

2. Ensure that all Managed services pods are in ready/running state, for example, 1/1.

- After installing the Managed services, **cam-provider-terraform-api** container fails to init.

The error **chown: /home/terraform: No such file or directory** is logged in the initContainer logs.

As a resolution, perform the following steps to install Managed services successfully:

1. Save the existing deployment in a file:

```
   oc -n cp4aiops get deploy cam-provider-terraform-api -o yaml > cam-provider-terraform-api-orig.yaml
```

2. Copy the saved deployment file to a new file:

```
   cp cam-provider-terraform-api-orig.yaml cam-provider-terraform-api-fix.yaml
```

3. Edit the **cam-provider-terraform-api-fix.yaml**

   - Remove **initContainers** section:

```
   initContainers:
      - args:
         - chown 1111:1111 /home/terraform && chmod 775 /var/camlog
         command:
         - /bin/sh
         - -c
         image: alpine:latest
         imagePullPolicy: IfNotPresent
         name: initcontainer
```

```
        resources: {}
        securityContext:
           allowPrivilegeEscalation: false
           capabilities:
              add:
              - CHOWN
              - FOWNER
              - DAC_OVERRIDE
              drop:
              - ALL
           privileged: false
           readOnlyRootFilesystem: false
           runAsNonRoot: false
           runAsUser: 0
           seLinuxOptions:
              type: spc_t
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
        volumeMounts:
        - mountPath: /var/camlog
           name: cam-logs
```

- Update the deployment name to **cam-provider-terraform-fix**:

```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
    annotations:
       deployment.kubernetes.io/revision: "2"
      name: cam-provider-terraform-api
```

4. Create the new deployment using **cam-provider-terraform-api-fix.yaml**:

```
   oc -n cp4aiops create -f cam-provider-terraform-api-fix.yaml
```

5. Restart IaaS pod after new cam-provider-terraform-api is ready:

```
   oc -n cp4aiops delete pod <cam-iaas-pod-instance>
```

6. Scale down the original **cam-provider-terraform-api deployment**:

```
   oc -n cp4aiops edit manageservice cam
```

7. Change the replicaCount to 0:

Note: Add the replicaCount under camProviderTerraform if the replicaCount is not added by default.

```
  camProviderTerraform:
   isolateRuntime: false
   replicaCount: 0
   runtime:
    replicaCount: 0
```

Troubleshooting issues in CLI tool

Known issues with Managed services CLI tool (camctl).

- If Managed services is installed on an OpenShift environment, the port value <CAM_PORT> defaults to 443. But the CLI login command in the **Configure the client** step shows the **cam_port** as **undefined**. The workaround is to replace **--cam_port undefined** with **--cam_port 443** and execute the **camctl login** command again. For example:

```
camctl login --cam_host <CAM_HOST> --cam_port 443 --username admin
```

Troubleshooting deployment issues

Some common issues, limitations, and logs related to deployment.

Common issues and solutions in IBM Cloud Kubernetes Service

The deployment of IBM Cloud Kubernetes Service template sometimes fails with a **404 page not found** error. It can occur when the cluster endpoints become temporarily unavailable in IBM Cloud during the deployment. Performing a plan and apply resolves the error.

Common issues and solutions in vSphere

- The following error might occur because of Self-Signed certificates:

  `provider.vsphere: Error setting up client: Post https://192.168.64.140/sdk: x509: cannot validate certificate for 192.168.64.140 because it doesn't contain any IP SANs`

  As a resolution, set the parameter "allow_selfsigned_cert" to "true".

- The following error occurs whenever vSphere virtual machine is in powered on state:

  `vsphere_virtual_machine.vm_1: The attempted operation cannot be performed in the current state (Powered on).`

  A virtual machine with same hostname/IP already present.

  - To create folder, use create_vm_folder=1.

    Use integer 1 to create a virtual machine only if it is not already available.

  - IPv4 Prefix: ipv4_prefix_length="24"

    This is a number string for prefix length.

Common issues in Amazon WebServiced Cloud (AWS)

- The following error message might occur whenever the virtual private cloud (VPC) is not found:

  `VPC not found`

  As a resolution, provide a "Name" for VPC and refer to the name in the template as opposed the VPC ID.

- Errors occur whenever you do not specify unique public keys.

- The following error might occur when using an older images for deployment:

  `Error: Your query returned no results.` You must change your search criteria and try again with **`data.aws_ami.aws_ami`**.

  The deployment using Ubuntu14 fails as AWS has removed support for older versions of operating system.

  The solution is to replace prefilled **`AMI`** with the available **`AMI`** in AWS. For example: replace **`ubuntu/images/hvm-ssd/ubuntu-trusty-14.04-amd64-server-*`** with **`ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*`**.

  The sample templates are also updated to use the latest **`AMI`**. To pull the latest updated templates, delete the existing **`templates/services`** and restart **`cam-iaas*`** and **`cam-service-composer-api*`** pods. Alternatively, import the updated sample templates/services, see Importing a template.

Common issues in IBM Cloud

- The following error message might occur because of SSH keys:

  `ibm_compute_ssh_key.orpheus_public_key: Error deleting SSH key: SoftLayer_Exception_Public: SSH key cannot be deleted because it is currently being used in an active transaction. (HTTP 500)Keys`

  IBM Cloud does not allow you to upload two identical pub keys having the same fingerprints. However, the IBM Cloud Terraform provider does not throw an error instead reuses that key. Issues occur whenever you try to destroy the key resource that is now associated with another running virtual machine.

  The solution is to use different keys for each deployment, reference an existing key or retry the destroy/delete to get past the error.

- The following error might occur when using an older image for deployment:

  `[ERROR] Error ordering virtual guest: SoftLayer_Exception_Order_Item_Unavailable: Debian GNU/Linux 9.x Stretch/Stable - Minimal Install (64 bit)`

  The deployment using Debian9 fails as IBM Cloud has removed support for older versions of the operating system.

  The solution is to replace prefilled Operating System ID with the available ID in IBM Cloud. For example: replace **`DEBIAN_9_64`** with **`DEBIAN_10_64`**.

  The sample templates are also updated to use the latest Operating System ID. To pull the latest updated templates, delete the existing templates/services and restart **`cam-iaas*`** and **`cam-service-composer-api* pods`**. Alternatively, import the updated sample templates/services, see Importing a template.

Common issues using older terraform engines or terraform templates

- The following error might occur when using deprecated terraform engine or terraform templates:

  NoCompatibleProviderEngineVersionError

  Terraform engine is removed and terraform templates are updated to use the Terraform 1.x.x version as Terraform engine 0.11.x is out of support.

  The solution is to use the latest terraform engine and terraform templates based on latest terraform version. For more information, see Managing Terraform Versions to use an older terraform version and see, Importing a template to import an older terraform template.

General issues

- Set a custom maximum number of Terraform jobs to process in parallel per cam-provider-terraform pod:

  1. Run the following command to edit the deployment for cam-provider-terraform.

     - If isolateRuntime = false, then run the following command:

       ```
       kubectl -n management-infrastructure-management edit deployment cam-provider-terraform-api
       ```

     - If isolateRuntime = true, then run the following command:

       ```
       kubectl -n management-infrastructure-management edit deployment cam-provider-terraform-runtime
       ```

  2. Update the value of MAX_LOCAL_TERRAFORM_JOBS.

     ```
     containers:
     - env:
       - name: MAX_LOCAL_TERRAFORM_JOBS
         value: "10"
     ```

     In this example, it is set to 10.

- The IP address might change after you restart a virtual machine, but it might not get reflected on the Managed services user interface. To refresh the resource state including the IP address, either run a plan/apply or use the refresh API to refresh the resource.

- Failures might occur in the deployment of a Terraform template that executes actions on a Windows image by using a WinRM connection type. As a resolution, check whether "AllowUnencrypted" parameter of WinRM configuration is set to true in the Windows image that is used in the Terraform template. For more information, see Terraform documentation.

- You might observe a failure whenever you do back-to-back interdependent VMware operations. For example, a "Start" followed by "Shutdown OS" causes a failure because the second action requires VMware Tools that has not got started yet.

- Managed services on IBM Cloud Pak for AIOps has a predefined 6 hours deployment window. Do the following steps to add **TERRAFORM_JOB_TIMEOUT_MS** environment variable to the deployment:

  1. Run the following command to open the deployment in edit mode.

     - If isolateRuntime = false, then run the following command:

       ```
       kubectl -n management-infrastructure-management edit deployment cam-provider-terraform-api
       ```

     - If isolateRuntime = true, then run the following command:

       ```
       kubectl -n management-infrastructure-management edit deployment cam-provider-terraform-runtime
       ```

  2. Add the **TERRAFORM_JOB_TIMEOUT_MS** variable with value.

     For example, the value of **TERRAFORM_JOB_TIMEOUT_MS** is updated to 10 hours in millisecond:

     ```
     name:TERRAFORM_JOB_TIMEOUT_MS
     value: "36000000"
     ```

- During template deployment, you must specify user credentials based on the operating system.

  - For IBM Cloud with Red Hat Enterprise Edition operating system, specify "root" as the user.
  - For AWS with Red Hat Enterprise Edition operating system, specify "ec2-user" as the user.
  - For AWS with Ubuntu operating system, specify "ubuntu" as the user name.

  If the appropriate user is not provided for the corresponding operating system, then the deployment fails with an error message.

  For example, If you do not provide "root" as the user while deploying a VMware template and the passwordless sudo is also not enabled, then the following error message is displayed:

```
Error: Response from pattern manager:
StatusCode:500
Message:
{
 "message": "Bootstrap command failed. See the pattern manager logs for more details.",
 "rc": 1,
 "request_id": "1c844be316b74601853d2ddde595d94d",
 "stderr": "Creating new client for cam-apache-1\nCreating new node for cam-apache-1\nConnecting to
0.0.0.0\nstty: 'standard     input': Inappropriate ioctl for device\nstty: 'standard input':
Inappropriate ioctl for device\nstty: 'standard input': Inappropriate   ioctl for device",
 "stdout": "0.0.0.0 knife sudo password: \nEnter your password: \r\n0.0.0.0 \r\n0.0.0.0 Sorry, try
again.\r \n0.0.0.0 knife sudo password: \n0.0.0.0 \r\n0.0.0.0 Sorry, try again.\r\n0.0.0.0 knife sudo
password: \n0.0.0.0 \r\n0.0.0.0 sudo: 3 incorrect password attempts"
}
```

- You can import templates from GitHub or GitLab and deploy them. However, if you remove or change access token in GitHub or GitLab and then try to destroy instance, the operation fails with the following error message:

  ```
  GithubClientError: vmware/terraform: failed to retrieve content list from github.; caused by
  {"message":"Not Found", "documentation_url":"https://developer.github.com/v3"}
  ```

  Ensure that the Managed services has a connection to GitHub/Gitlab when you destroy the instance.

- **CAM deployment stays in Progress - Log File repeating null_resource.clone_git: Still creating..."**

  The log file can record the template deployment status as in progress for a very long time. To identify the reason for the issue, do the following analysis on the deployed virtual machine wherein the problem occurred:

  - Run the following command to check whether your user has passwordless sudo:

    ```
    sudo cat /etc/sudoers
    ```

    When it prompts for a password, enable passwordless sudo for the specific user.

  - If the user does not have sudo privileges, add that user to **/etc/sudoers**.

  - Run the following command to check if name server is set up in your machine:

    ```
    ping github.com
    ```

  - Check if the curl or Git commands are installed on the virtual machine. It might be an issue with the apt mirror.

- SSL Certificate error while deploying a template.

  To setup Git client locally:

  - Create **.gitconfig** file in the same Git repository as the template. The file should point to the correct CA certificates.

    ```
    [http]
        sslCAInfo = /home/terraform/certs/ca-bundle.crt
    ```

  - When the provider Terraform deploys the template, it downloads the template and all its files from the Git repository, which includes the **.gitconfig**. When the Terraform init is executed, the Git client looks up in the current folder (user's home folder for the current process) and it uses the appropriate CA certificates to download the modules from Git.

- To debug Terraform template deployment, set **TF_LOG** environment variable in **cam-provider-terraform** pod. You can set **TF_LOG** to one of these log levels **TRACE**, **DEBUG**, **INFO**, **WARN**, or **ERROR** to change the verbosity of the logs. **TRACE** is the most verbose log level. This results in the detailed logs to appear on **stderr**.

  1. Run the following command to open the deployment in edit mode:

     - If isolateRuntime = false, then run the following command:

     ```
     kubectl -n management-infrastructure-management edit deployment cam-provider-terraform-api
     ```

     - If isolateRuntime = true, then run the following command:

     ```
     kubectl -n management-infrastructure-management edit deployment cam-provider-terraform-runtime
     ```

  2. Add the **TF_LOG** variable with value.

     ```
     name: TF_LOG
     value: "TRACE"
     ```

Logs

The logs that you can examine are as follows:

- Check the template instance Terraform logs on the UI for Terraform or template errors.
- Check logs in containers for internal Managed services errors wherein the **`CAM_logs`** is in the **`cam-logs-pv`** path:
    - **`/CAM_logs/cam-iaas`**
    - **`/CAM_logs/provider-terraform-local`**

Troubleshooting security related issues

Some known security related issues in Managed services.

- If you deploy multiple template instances with the same SSH key content to the IBM Cloud environment, an issue occurs when trying to delete the instances.
- You can only view the deployments (template instances and service instances) that are allowed for the specified namespace. **`Namespace`** dropdown allows you to switch between the available namespaces and to see the deployments of those namespaces.

  Note: If your user role is not authorized to perform a certain action, a 401 (unauthorized) error occurs in the API. However, the error is not displayed in the user interface. Symptoms may include buttons that do not respond to clicks or the page redirects in the user interface. For example, When you click deploy, the page gets redirected back to the list of instances without performing deployment. If you are logged in as an LDAP user, then go to IBM Cloud Pak for AIOps user interface and ensure that the user is assigned to a team that has access to one or more namespaces.
- Examine the following logs for security related issues:
    - cam-iaas
    - cam-orchestration
    - cam-portal-api

  Note: LDAP is configured in the IBM Cloud Pak for AIOps user interface. You can use roles in Managed services only if the LDAP is configured.

Open source support in Managed services

Because Managed services contains open source software, IBM offers some support for the open source software that it uses.

IBM uses open source software in three ways:

- To help its valued customers avoid vendor lock-in
- To ensure compatibility with the broadest ecosystem of software
- To accelerate development of its products

The mature, enterprise-level open source components that are used in Managed services was carefully selected, integrated, and tested to work as described in the product documentation and licenses. The components include integral open source components like Terraform, Kubernetes, Docker Engine, Helm, and ELK stack.

The IBM Software Support Handbook is the definitive reference for IBM's open source support statement. To review IBM's open source support statement, see Third party software and Open Source software in the IBM Software Support Handbook. The support team for Managed services uses the information in this statement to scope its interactions with you.

The main goal of the Managed services support team is to help paying customers get their supported product functional again as quickly as possible. Although the community support options are available for all Managed services users, contacting IBM Support is the best support option for users of the different bundled enterprise editions of Managed services. The community support channels are the only support options forManaged services Community Edition users.

When you contact IBM support, there are two phases of open source support:

1. The problem determination phase. During this phase, support engineers determine whether the root cause of an issue is due to open source code or IBM code.
2. The open source support phase. If the root cause is due to open source code, support engineers identify which open source project contains the issue and work with you to identify next steps.

If a critical defect is found in the open source software within Managed services, IBM engineers use open source community resources to deliver a fix to the problem. The resulting fix is integrated, tested, and released to customers as an emergency update and in the next product release. If no fix is available from the community for a critical defect, IBM might use reasonable commercial efforts to provide a test fix to customers and then work with the specific open source community to create an official fix. The final arbiter over whether a supported fix can be provided belongs to the open source community.

Note that if any enterprise-level open source project has a critical defect, many clients are affected, and the community works as fast as possible to resolve it. It is unusual for only one customer to encounter a critical open source defect.

IBM cannot support any open source software that is not packaged with Managed services. In addition, if a client chooses to make any additions, subtractions, or upgrades to the Managed services open source code outside of product updates, that code is not supported. For Cloud Foundry, IBM might assist you with reporting defects in community buildpacks and runtimes to the appropriate open source community. IBM cannot provide fixes for these Cloud Foundry community buildpacks and runtimes.

Note: Managed services will support new versions of supported operating systems, Kubernetes, Docker, and other dependent infrastructure after new releases happen and when they are fully tested by the Managed services team.

Importing custom CA Certificate into Terraform provider pod

Your Terraform template might include Terraform modules whose source might be in a source control management system that is secured by your organization's internal certificate. To avoid certificate errors while making calls to Terraform modules follow these steps to add your own CA certificate chain to the existing list of pre-configured certificates.

1. Run the following command to get the name of the pod that runs the Terraform engine:

   ```
   kubectl -n cp4aiops get pods | grep cam-provider-terraform
   ```

   If this command returns pods that are prefixed with **cam-provider-terraform-runtime** and **cam-provider-terraform-api**, then you are running Terraform engine in isolation mode. In this mode, the Terraform runtime runs in separate pod. If this command returns pods that are prefixed only with **cam-provider-terraform-api**, then you are running the Terraform engine in non-isolation mode.

   If Terraform engine is deployed in an isolation mode, then you must apply the rest of the steps to **cam-provider-terraform-runtime** pod and deployment else apply the steps to **cam-provider-terraform-api** pod and deployment.

2. Run the following command to copy the existing CA bundle file from a Terraform engine pod.

   ```
   kubectl -n cp4aiops cp <terraform-engine-pod-name>:/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem ca-bundle.crt
   ```

   Where **<terraform-engine-pod-name>** is:

   - For **Isolation mode**: A pod name prefixed with `cam-provider-terraform-runtime`
   - For **Non-isolation mode**: A pod name prefixed with `cam-provider-terraform-api`

   You might see the following error, ignore it and check whether the file got copied.

   ```
   tar: Removing leading `/' from member names
   ```

3. Add your CA certificate to the copied CA bundle file **ca-bundle.crt**.

   1. Back up the original CA bundle file **ca-bundle.crt**.

      ```
      cp ca-bundle.crt ca-bundle-original.crt
      ```

   2. Add your CA certificates (root and any intermediary) to the **ca-bundle.crt** file.

      ca-bundle.crt file has number of certificates as follows:

      ```
      -----BEGIN CERTIFICATE-----
      <certificate_content>
      -----END CERTIFICATE-----

      -----BEGIN CERTIFICATE-----
      <certificate_content>
      -----END CERTIFICATE-----
      ```

      Append your certificates at the end of the file as follows:

      ```
      -----BEGIN CERTIFICATE-----
      <certificate_content>
      -----END CERTIFICATE-----

      -----BEGIN CERTIFICATE-----
      <certificate_content>
      -----END CERTIFICATE-----

      -----BEGIN CERTIFICATE-----
      <my_custom_certificate_content>
      -----END CERTIFICATE-----
      ```

      If you have your certificate in a file, then you can execute the following command to append your certificate to **ca-bundle.crt**

```
cat <your-ca-cert-file> >> ca-bundle.crt
```

4. Create a secret from a modified `ca-bundle.crt` file.

   ```
   kubectl create secret generic cam-custom-cert-file --from-file=./ca-bundle.crt -n cp4aiops
   ```

5. Patch the Terraform engine pod deployment file to mount a new volume that contains the created secret file.

   a. Create a file `cam-provider-terraform-patch.json` with the following contents:

   ```
   {"spec": {"template": {"spec": {"containers": [{"name": "< terraform-engine-deployment>","volumeMounts": [{"mountPath": "/etc/pki/tls/certs","name": "cam-custom-cert-bundle","readOnly": true}]}],"volumes": [{"name": "cam-custom-cert-bundle","secret": {"defaultMode": 420,"secretName": "cam-custom-cert-file"}}]}}}}
   ```

   Where `<terraform-engine-deployment>` is:

   - For **Isolation mode**: `cam-provider-terraform-runtime`
   - For **Non-isolation mode**: `cam-provider-terraform-api`

   b. Run the following command to patch the deployment to mount the new CA bundle to Terraform engine pods.

   ```
   kubectl -n cp4aiops patch deployment <terraform-engine-deployment> --patch "$(cat cam-provider-terraform-patch.json)"
   ```

   Where `<terraform-engine-deployment>`:

   - For **Isolation mode**: `cam-provider-terraform-runtime`
   - For **Non-isolation mode**: `cam-provider-terraform-api`

Troubleshooting user interface

Known issues in the Managed services user interface.

- When all Managed services pods are running, but you cannot still log in to the user interface, then it might be an issue with oauthcache.
- The plan functionality fails to work whenever you do the following steps:

  1. Return to the **modify** tab of a deployed instance wherein a plan is already run.
  2. Click **Back** or **Edit parameters**.

  As a workaround, do the following steps:

  1. Click select version step.
  2. Edit parameters page.
  3. Click **Plan** again to start the plan process
- The following error might occur when you start the console in spite of Managed services services running in IBM Cloud Pak for AIOps:

  ```
  CWOAU0061E: The OAuth service provider could not find the client because the client name is not valid.
  ```

  To resolve the problem, run the following commands to restart the cam-iaas pod and wait for the pod to start:

  ```
  kubectl get pods -n cp4aiops | grep iaas
  kubectl delete po/<pod_name> -n cp4aiops
  ```

  Wait for the pod to start. Run the following command to confirm whether it is in running state:

  ```
  kubectl get pods -n cp4aiops | grep iaas
  ```

- If you are redirected to the getting started page in Managed services user interface after the restart of IBM Cloud Pak for AIOps or you encounter other user interface related issues, then restart the user interface microservices. Run the following command to restart the user interface pods:

  ```
  kubectl delete -n cp4aiops -l 'name in (cam-portal-ui,cam-service-composer-ui,cam-ui-basic,cam-ui-connections,cam-ui-instances,cam-ui-templates)' pods
  ```

- As an LDAP user you might get redirected to the getting started page. To avoid such erroneous redirection, go to IBM Cloud Pak for AIOps user interface and ensure that the user is assigned to a team that has access to one or more namespaces.
- There might be performance issues in Managed services user interface whenever you use Mozilla Firefox browser. If this issue affects all websites wherein you use self-signed certificates, then the problem might be because of SSL certificates. If a service generates a new certificate for a particular website every time it is restarted, then Mozilla Firefox stores all these certificates in its local database and verifies whether the path is valid. Multiple self-signed certificates of the same name can slow down the turn around time of your browser. In addition, the TLS handshake operation ceases to respond. To identify the root cause of the problem and resolve it temporarily, open the profile folder of your Mozilla Firefox browser and locate the `cert8.db` file. Replace the file with a different name and revisit the same site

that had problems. If the page displays properly, then the issue is because of local certificate database having too many self-signed certificates of the same name. Do the workaround often to avoid problem recurrence.

- Turn on the inspect for the browser and look for any errors returned on the API. For example, an API call might return a 401 (unauthorized), or might return a 400 (bad input) with an error message in the response, which is not displayed on the API. Page redirects, looping/refreshing pages might indicate a bad API call being returned to the user interface.

  To debug such errors, examine the following logs:

  - cam-ui-basic
  - cam-ui-connections
  - cam-ui-instances
  - cam-ui-templates
  - cam-portal-ui
  - cam-service-composer-ui
- The logout button might not be visible with some screen resolutions or if you zoom in the Managed services user interface. To solve the problem zoom out the user interface or access directly the logout page at **`https://<CAM_IP_address>:30000/logout`**.
- The Internet Explorer browser is not supported.
- Managed services UI shows one of the following errors:

  - Topology destroyed error while deployment
    - To resolve this issue, check the logs of **`cam-orchestration`** and if there are any errors, restart the pod.
  - Pool destroyed error on UI while accessing Managed services
    - To resolve this issue, check the logs of **`cam-ui-basic`** and if there are any errors, restart the pod.
  - Internal server error on UI while accessing Managed services
    - To resolve this issue, check the logs of **`cam-portal-ui`** and if there are any errors, restart the pod.

  Check whether all the pods are up and running for namespace **`cp4aiops`**.

Troubleshooting Service Composer issues

Some known issues that are related to Service Composer or Portal-related issues in Managed services.

- If the logged in user is not able to import services from GitLab and the following error occurs, then it indicates that the user does not have access to Helm repositories:

  ```
  "Activity_id: ibmchartc7e3fb29, The template ibm-nodejs-sample does not exist."
  ```

  As a resolution, grant access to Helm repository for the logged in user and then import service from GitHub/Gitlab/Bitbucket Server.

- When you use special characters in JSON values that are used as an input parameter for a template, the provisioning fails with an error from IaaS.

  For example,

  ```
  "amis": { "today": "\"Thursday, January 24, 2019 11:01 AM\"" }
  ```

  As a resolution, remove the special characters and retry provisioning. The following example is cleared off special characters:

  ```
  "amis": { "today": "Thursday, January 24, 2019 11:01 AM" }
  ```

- For some reason, if a service instance is stuck in IN_PROGRESS state for a long time without any backend activity, then delete the service instance entry from the database. This situation might arise because of an unplanned restart of any of the Managed services pods or any other related issues. Use Cleanup Service Instance in inprogress state API to delete the service instance. The API section provides you information about any **`ResourceLockError`** that might occur during the cleanup process.

- If you do not see any content on the palette of the composer tab, then it is because of the connectivity issue between service composer and Business Process Manager or IBM Cloud Orchestrator or Helm plug-ins.

  IaaS sample error message:

  ```
  Bad return code from provider: { statusCode: 500, body: '{"error":{"statusCode":500,"message":"Internal Server Error"}}' }
  ```

  Business Process Manager Plugin/Provider Error

  ```
  [2019-05-08T06:29:47.422] [ERROR] http_utils - { Error: connect ETIMEDOUT <IP_ADDRESS>:9443 at TCPConnectWrap.afterConnect [as oncomplete] (net.js:1191:14) errno: 'ETIMEDOUT', code: 'ETIMEDOUT', syscall: 'connect', address: '<IP_ADDRESS>', port: 9443 } Error: Error stack Logged at: at Logger.(anonymous function) [as error]
  ```

As a workaround, disable Business Process Manager plug-in that is causing the failure. For the actual procedure to disable a plug-in, see Enabling and disabling Business Process Manager and IBM Cloud Orchestrator.

- Even after the service is deployed and is in a running state, the IBM Cloud Pak for AIOps or IBM Kubernetes Services cluster does not show up in the controller's cluster list.

  It is possible that the service deployment had taken long enough that the controller token that is passed as a parameter on service deployment has been refreshed since the service deploy action. In this case, the IBM Cloud Pak for AIOps cluster is unable to register to the controller. To verify that this is the issue, check that the token on the controller instance has the same value as the one you have used when you deployed the Managed services. As a workaround for this issue, try any of the following methods:

  - Method 1:

    1. Redeploy the service.
    2. Ensure that the controller token stays unchanged for the time it takes for the service to deploy the new cluster.
    3. Attach the token to the controller.
    4. Estimate the duration of service deployment.
    5. Set the refresh of the controller token beyond that time interval.
  - Method 2

    1. Deploy the new cluster by using the Managed services templates.
    2. Use `Loads_Images_and_Deploys_Chart_on_ICP` service to install the chart.
    3. Register the cluster with the controller. Use `Services/MCM/MCM_on_ICP/MCM_Klusterlet-Loads_Images_and_Deploys_Chart_on_ICP/MCM_Klusterlet-Loads_Images_and_Deploys_Chart_on_ICP.json` from Git repository.

    Using this two-step cluster registration has the benefit that the long running step, which is the deployment of the cluster, gets separated from the chart install and connect operation. As the controller token is passed through during the service deploy, and this service deploy takes only minutes to deploy and run, it is unlikely for the token to be invalidated during the service deployment. If the issue still persists, redeploy just the chart service using the new token, rather than redeploying the entire cluster again.

- When you keep multiple browser instances open for the same deployed instance (instance details or associated instance) that is in "wait" state, manually refresh the browser to see the latest updates.

- Service may fail with unexpected EOF or may fail with both connection timeout and unexpected EOF. As a workaround, do the following steps:

  Set the `TERRAFORM_PARALLELISM_COUNT` environment variable on the running cam-orchestration deployment of Managed services:

  - Using the user interface:

    1. Login to IBM Cloud Pak for AIOps user interface and go to **Workloads** > **Deployments**.
    2. Search for `cam-orchestration` deployment.
    3. Select edit from the overflow menu.
    4. Edit the deployment by adding the following element to the `env:` array and then click **Submit**.

       ```
       {
        "name": "TERRAFORM_PARALLELISM_COUNT",
        "value": "1"
       }
       ```

    5. To verify whether your deployment is successful, double-click `cam-orchestration` deployment link.
    6. In the **Overview** tab > **Pods** section, check whether the `cam-orchestration` pod has restarted successfully.
  - Using the command line:

    1. SSH to the Managed services deployment machine.
    2. Run the following command to edit the `cam-orchestration` deployment:

       ```
       kubectl edit deployment cam-orchestration -n cp4aiops
       ```

    3. Under the `- env:` section, add the following lines.

       Note: Make sure you maintain the indentation same as other name-value pairs in this section.

       ```
       name: TERRAFORM_PARALLELISM_COUNT
       value: "1"
       ```

    4. Run the following command to ensure that the cam-orchestration pod has come up successfully after the change:

       ```
       kubectl get pods -n cp4aiops | grep cam-orch
       ```

5. Save these changes.

- When you deploy a service that has non-permissible special characters in the value of any of its parameters, then the service instance creation fails with an error. For example, a service deployment fails because of a period (.) in the key value of a MongoDB parameter.
- Mongo does not allow you to create a map with a key value having dot in the name. To resolve this Mongo limitation, use a list with key names that have hardcoded text without a dot.
- If you use the same bind secret across multiple provisioned instances, the first bind instance is created correctly. From the second instance onwards, though a bind instance gets created, the following error occurs in IBM Cloud Pak for AIOps user interface:

  ```
  ServiceBroker returned failure; bind operation will not be retried: Status: 400;
  ErrorMessage: <nil>;
  Description: <nil>;
  ResponseError: <nil>;
  ```

- When you create a service instance with bind action in IBM Cloud Pak for AIOps user interface, you cannot terminate the instance from IBM Cloud Pak for AIOps user interface without removing associated bind instances. Though you can terminate it from Managed services user interface, the bind action remains in active state.
- For IBM Cloud Kubernetes Service template in Service Composition, do not select or enter values for datacenter and other networking attributes whenever the value of the **`machine_type`** is "free". This selection is not supported because the datacenter parameter along with other networking attributes are ignored by the Terraform provider for free clusters.
- If you are logged into Managed services for a very long time, then its Access-Token on the browser expires. If you try to deploy a Service Instance after the Access-Token expires, the deployment fails with a 401 error. As a resolution, do the following steps:

  1. Logout of Managed services.
  2. Relogin to Managed services.
  3. Retry service instance deployment.
- When you drag-drop Helm charts in the composer, parameter details are not fetched properly. To resolve this error, synchronize the IBM Cloud Pak for AIOps Helm repositories in IBM Cloud Pak for AIOps user interface. For more information, see Managing Helm repositories.
- Whenever an email notification is sent, the SMTP user name value is also used as the sender email address value. However, sometimes when you use services like sendgrid, the SMTP user name is not a valid email address. Though send a test email results in a message that says **`Test email sent successfully.`**, no email is actually sent or received.
- Terminate service instance may go to error state wherever the service instance has more than one Terraform template /Helm chart or is a combination of Terraform and Helm chart. A deployment failure in a template or helm chart can cause the entire terminate service instance to fail. However, the success of the destroy action of individual template or helm charts is based on their sequence in the composition.

  For example, consider the following behavior whenever a Helm chart and Terraform template are placed in sequence and a failure occurs in Helm chart deployment:

  - If Terraform template is positioned after the Helm chart in the sequence, it gets destroyed but the termination fails at the Helm chart. The rest of the sequence is not tried and the service instance termination fails.
  - If the Helm chart is positioned after the Terraform template in the sequence, it is never attempted as the Helm failure occurs before the destroy action of Terraform template.

  To resolve this issue, clean up Helm charts from backend. For the Terraform template, you can clean from backend or from the template tab.
- Helm chart limitations:

  - If you observe the following error message in the activity logs, then ensure the helm tiller has been configured on your IBM Kubernetes Service cluster:

    ```
    Error: could not find tiller
    ```

  - Helm deployments assume that you have precreated the required persistent volumes, otherwise it uses dynamic provisioning (for example GlusterFS).
  - When you click the link for helm release in the **View a deployed service instance** page, you are forwarded to the IBM Cloud Pak for AIOps management console, and not to the **Deployed templates** page in Managed services.
  - Helm release deployments are not visible from the deployed templates list view in Managed services. They are only visible from the **Service instance details** page and IBM Cloud Pak for AIOps Helm releases page.
- When you rectify an invalid primary email server and redeploy a service that is already deployed, email notification may fail with an error message.

  An example error message is as follows:

  ```
  serviceblueprint_email.emailnot379ed899: Creating...
  connection_used: "" => ""
  fail_quietly: "" => "true"
  ```

```
message: "" => "Test"
sender_name: "" => "CAMadmin"
sequence_no: "" => ""
status: "" => ""
status_error_message: "" => ""
status_message: "" => ""
subject: "" => "Test"
to.#: "" => "1"
to.0: "" => "abc@in.ibm.com"
Error applying plan:

1 error(s) occurred:

serviceblueprint_email.emailnot379ed899: 1 error(s) occurred:
serviceblueprint_email.emailnot379ed899: unexpected EOF
Terraform does not automatically rollback in the face of errors
....
....
```

Do the following steps to resolve the error:

1. Go to IBM Cloud Pak for AIOps console and click **Automate infrastructure>Managed services**.
2. Delete primary and secondary SMTP connection.
3. Open new tab in same browser without closing the browser.
4. Open any REST Client extension in the browser and call the following API:

   ```
   URL : https://<TSA_HOSTNAME>/cam/connections/call_proxy?
   service=composer&httpmethod=POST&relative_url=smtpconnections
   Header: Content-Type : application/json
   Method: POST
   Payload:
   ```

   Payload For Primary connection:

   ```
   {
     "name": "PRIMARY",
     "description": "Primary SMTP Connection",
     "connection_parameters": [
       {
         "name": "username",
         "value": "<primary@somesmtpserver.com>"
       },
       {
         "name": "password",
         "value": "<password>"
       },
       {
         "name": "hostname",
         "value": "<smtp.somesmtpserver.com>"
       },
       {
         "name": "port",
         "value": "<587>"
       }
     ]
   }
   ```

   Payload For Secondary connection:

   ```
   {
     "name": "SECONDARY",
     "description": "Secondary SMTP Connection",
     "connection_parameters": [
       {
         "name": "username",
         "value": "<secondary@somesmtpserver.com>"
       },
       {
         "name": "password",
         "value": "<password>"
       },
       {
         "name": "hostname",
         "value": "<smtp.somesmtpserver.com>"
       },
       {
         "name": "port",
         "value": "<587>"
   ```

```
    }
  ]
}
```

- If you remove the mapping of service parameters in the **Composition** tab > **Parameters** tab of the template activity, it does not delete it from the main **Parameters** tab > **Service Parameters** section of the service and is also available while you create a service instance. If you do not want to see it while you create a service instance, you must delete it exclusively from the main **Parameters** tab > **Service Parameters** section.
- There might be failures in the termination of a template instance in spite of a "Terminated" message in the user interface. After you terminate a service instance, check the template instance logs to reconfirm.
- Whenever an error occurs during category creation, check for API-related errors in the log file.
- When you order a service, do not map the same parameter to two different decisions in the Additional Parameters page. This causes both decisions to follow the same path.
- If you do not provide the instance name in the composition tab for a template, then an error message is displayed in the **Source Code** tab during publish. To avoid such errors, whenever you add a template in the composition tab, provide a value for the instance name.
- Examine the following logs for Service Composer or Portal-related issues:
  - cam-orchestration
  - cam-portal-api
  - cam-iaas
- If Managed services orchestration has multiple templates and a template fails in the workflow, then all subsequent templates are also not considered for deployment.

Troubleshooting Service library issues

Some known issues that are related to the Service library.

- When Managed services is reinstalled, the following error is seen when Service library is accessed:

  `Error Connection failed to fetch deployable services.`

  To resolve this issue, restart **`service-library-ui-api`** pod by using the following command:

  `oc delete pod <service-library-ui-api-xxx> -n cp4aiops`

- When default services do not show up in Service library, delete the **`cam-service-composer-api`** pod to reinitiate the call to pull the services from GitHub. Delete the pod by using the following command:

  `oc delete pod <cam-service-composer-api> -n cp4aiops`

- If the **Service library** menu entry is missing from the navigation menu **Automate infrastructure** for the first time, try accessing the Service library by using the following URL: **`https://<cp4mcm-console>/servicelibrary`**. Then, the Service library entry is visible in the navigation menu.

Troubleshooting middleware content

Common issues in middleware content.

For more information about common issues or errors that may occur in middleware content, see the following troubleshooting references:

- For common issues while deploying the Content Runtime, see Troubleshooting and known issues.
- For issues while populating the Software Repository, see Populating the software repository.
- For issues while deploying Middleware Templates, see Troubleshooting issues and useful links.
- For issues while deploying customizing Middleware Content, see Customizing a predefined template.

In addition to these references, you can also refer to the following logs:

- Content Runtime - Pattern Manager logs, go to **`/var/log/ibm/docker/pattern-manager/flask.log`**.
- Content Runtime - Software Repository logs, go to **`/var/log/ibm/docker/software-repo/var/nginx/log`**.
- Content Runtime - Chef Server logs, go to **`/var/log/opscode/*`**
- Deployed Virtual machine, go to **`/var/log/chef-client.log`** that are displayed in the UI logging.

Troubleshooting content related issues and useful links

Learn how to isolate and resolve problems with the Managed services Middleware Content Runtime.

The following logs are useful in determining deployment failures:

- Middleware product logs for installation and configuration issues
- Template deployment instance logs
- Chef client logs
  - `/var/log/chef-client.log`
- Chef server logs
  - `/var/log/opscode`
- Content Runtime Pattern Manager log (on Content Runtime Instance)
  - `/var/log/ibm/docker/pattern-manager/flask.log`

If you have a question about Managed services Content Middleware templates, check the dedicated Stack Overflow channel to see whether it is already answered. If not, post your question and it will be answered by a member of the Middleware Template development team or the wider community.

Known restrictions

- When you correct a public key of content runtime and redeploy the middleware template, the template deployment might still fail due to key pair mismatch. As middleware deployments do not update content runtime variables on plan and apply, deployment fails even if you destroy (not delete) the middleware template and use plan and apply to recreate it.
- Managed services content does not configure yum repos. Ensure that a correct configuration is set in the image template.

Useful links

- IBM WebSphere Application Server IBM Documentation
- IBM Db2 Documentation
- IBM MQ Documentation
- Chef Documentation

Template deployment failures

- Starter Library templates are not loaded
- SSH handshake failure
- Terraform provider does not support resource
- SSH key not found
- Node already registered with the Chef server
- Failure cleanup does not occur
- Known issues with VMware vSphere
- Recovering the Content Runtime
- Bootstrap fails with knife sudo password error
- Troubleshooting issues in IBM Cloud Pak for AIOps or Red Hat® OpenShift® Container Platform template deployments

Starter Library templates are not loaded

Some of the Starter Library templates may not load properly in Managed services because of invalid git token.

To fix this issue, run the steps mentioned in the Configuring Managed services to import out of box templates and services by using your github.com token.

SSH handshake failure

SSH handshake failure occurs when the credentials supplied don't match the operating system credentials when creating the virtual machine.

Symptoms

When the virtual machine image is started, the bootstrap mechanism for the Terraform cloud provider needs the ability to log in to the virtual machine, but the operating system credentials were not correct.

```
Error applying plan:
1 error(s) occurred:
* ibmcloud_infra_virtual_guest.referenced-image: 1 error(s) occurred:
* ssh: handshake failed: ssh: unable to authenticate, attempted methods [none publickey], no supported
methods remain
```

Resolving the problem

- For IBM Cloud and AWS:

When deploying an IBM Cloud or AWS template, you specify the advanced content runtime to use for the deployment. The VM is created using the AWS key pair and private key for pattern manager that was specified for that advanced content runtime. If you are seeing the SSH handshake failed error message, it is likely that the private key for pattern manager is not the correct private key for the key pair defined in IBM Cloud or AWS.

Using the process described on Content Runtime plan and apply, modify the advanced content runtime by updating either the Key Name for Pattern Manager key set field or the private and public keys for pattern manager fields. Once the apply completes, you will be able to deploy your template to the advanced content runtime.

- For VMware:

Verify that the correct operating system user name is provided in the Terraform template. To determine the default user name, see the cloud provider documentation.

Terraform provider does not support resource

The Terraform provider cannot find a resource that is defined in the Middleware Terraform template.

Symptoms

The following error is displayed:

```
1 error(s) occurred:
* camc_notaresource.referenced-image_db2_v111_install: Provider doesn't support resource: camc_notaresource
```

Resolving the problem

Edit your Terraform template and find the resource. Verify that it is one of the following values and correct the misspelled name:

- `camc_softwaredeploy`
- `camc_bootstrap`
- `camc_vault`

SSH key not found

During provisioning of the middleware template, data lookups are being performed to retrieve information from the IBM Cloud or Amazon Web Services (AWS) accounts. When this error is encountered, the Pattern Manager public SSH key cannot be found in the appropriate vendor SSH key list.

Symptoms

If this key cannot be located, the Pattern Manager is not able to install the public SSH key on the virtual machine.

```
* data.ibmcloud_infra_ssh_key.camc_public_key: data.ibmcloud_infra_ssh_key.camc_public_key:
No ssh key found with name [camc_public_ssh_key_name]
```

Resolving the problem

The **camc_public_ssh_key** must be recreated in the appropriate Cloud account that is being used by the Managed services Automation Content runtime.

The **camc_public_ssh_key** was provided during the deployment of the Managed services Automation Content template deployment.

- For IBM Cloud:

  1. Log in to the IBM Cloud Console.
  2. Select **Classic Infrastructure** > **Manage** > **SSH Keys**.
  3. Click **Add**.
  4. Cut and paste the contents of the **camc_public_ssh_key** into the **Key Contents** section.
  5. Enter the name of the key in the **Label** section.
  6. Click **Add**.

- For AWS:

  1. Log in to the Amazon Web Services console.
  2. Click **All services > Compute > EC2**.
  3. Under the **NETWORK & SECURITY** section, click **Key Pairs**.
  4. Recreate the **camc_public_ssh_key** by clicking **Create Key Pair** or **Import Key Pair**.

Node already registered with the Chef server

During the provisioning of each virtual machine resource in the middleware template, the Managed services Pattern Manager registers the virtual machines short host name with the Chef server. This registration adds the host name as a Chef node.

Symptoms

The pattern manager throws the following error when the Chef node already exists:

```
1 error(s) occurred:
* camc_bootstrap.referenced-image_chef_bootstrap_comp: 1 error(s) occurred:
* camc_bootstrap.referenced-image_chef_bootstrap_comp:
**********
Error: Response from pattern manager:
StatusCode:400
Message:
{
  "message": "Node huhnHostName is already registered with the chef server"
}
**********
```

Causes

This issue can occur in the following scenarios:

- If the same host name was specified more than once during the deployment of the template.
- If you deploy an instance of the template and specify a host name that is already registered with the Chef server.

Resolving the problem

The solution for the issue depends on the cause of the duplicate Chef node names. Perform one of the following actions that corresponds to the cause of the issue:

- Make sure that the middleware template that is being deployed has unique host names for each virtual machine (node) that is being provisioned. If it does not, delete the deployed instance, correct the template, and deploy again.
- Pick a unique host name that is not being used by an already deployed middleware template instance. If you have access to the Chef server, you can run the following command to list all nodes that are registered:

  ```
  knife node list
  ```

- If you are sure that you are using a unique host name for all middleware template deployed instances, you can delete the Chef node from the server. This action should only be done if you are certain that the Chef node is no longer being used by another middleware template deployed instance.

  Log in to the Chef server and delete the node:

  ```
  knife node delete [Node_Name]
  ```

Failure cleanup does not occur

Pattern Manager does not do the failure cleanup after a software deployment failure.

Symptoms

The software deploy process involves the following steps:

1. Update vault.
2. Update runlist.
3. Update node attributes.
4. Do software deployment.

If the software deployment fails, the items created during the previous three steps are not removed.

Resolving the problem

If you have access to a Chef server, you can run the following commands:

1. Update or delete vault:

   ```
   knife vault update [vault] [item] '[value]'
   ```

   or

   ```
   knife vault delete [vault] [item]
   ```

2. Update runlist:

   ```
   knife node run_list set [node_name] [runlist]
   ```

3. Update node attributes:

   Use **`knife exec`** to remove node attributes.

   Note: **`knife exec`** is a powerful tool and it should be used carefully.

Known issues with Managed services and VMware vSphere

Read the following list of known issues with Managed services deploying to VMware vSphere.

Incorrect credentials for the operating system template

The credentials specified in the operating system user name and operating system password during a deployment need to be set to valid credentials for the operating system template that you are using for the deployment. If they do not match, the deployment request fails with the following error:

```
Error applying plan:

1 error(s) occurred:

* vsphere_virtual_machine.DB2Node01: 1 error(s) occurred:

* ssh: handshake failed: ssh: unable to authenticate, attempted methods [none password], no supported methods remain
```

Enter the correct credentials for the operating system template and redeploy your instance.

Multiple virtual machines with the same IP address

When deploying a virtual machine, there is no checking that the specified IP address is already in use by a different virtual machine on your vSphere instance, so an IP address conflict within your network might occur. IP address management should be handled outside of IBM Managed services.

Virtual machines can be reused

Terraform reuses an existing VM in your VMware vSphere environment if the following conditions are met:

- The short hostname matches an existing virtual machine in vSphere.
- The existing virtual machine in vSphere is stopped.

If the existing virtual machine was deployed to the same Content Runtime as your current deployment, the deployment fails with the error that the node is already registered with the Chef server. For more information, see Node already registered with the Chef server.

If the existing virtual machine was deployed to a different Content Runtime as your current deployment, the deployment may declare that it completed successfully. However, the virtual machine is now deployed to the new Content Runtime Chef server and is no longer registered to the old Content Runtime Chef server. Any recipes or roles from the old Chef server is no longer applied to this virtual machine. It is recommended to always deploy to a new virtual machine.

The attempted operation cannot be performed in the current state (Powered on)

This issue occurs when you have specified a short host name that matches an existing virtual machine in vSphere. The following error is displayed, for example:

```
Error applying plan:

1 error(s) occurred:

* vsphere_virtual_machine.MQV9Node01: 1 error(s) occurred:

* vsphere_virtual_machine.MQV9Node01: The attempted operation cannot be performed in the current state (Powered on).
```

Choose a different short hostname and redeploy your instance.

Virtual machine not found

Terraform raises the following issue if it cannot find the expected virtual machine in your VMware vSphere environment. This issue typically happens if vSphere had a problem creating the VM but did not report the failure in an expected fashion.

```
Error applying plan:

1 error(s) occurred:

* vsphere_virtual_machine.HTTPNode01: 1 error(s) occurred:

* vsphere_virtual_machine.HTTPNode01: vm 'managed_services/content/content-dev63' not found
```

In the previous error, **`Content`** is the vSphere folder name and **`content-dev63`** is the short hostname specified during deployment. Check the vSphere task log files to see why the virtual machine creation failed.

Recovering the Content Runtime

Learn how to recover your Managed services environment after losing the Content Runtime instance.

If the Content Runtime virtual machine is lost with no backup, there is a path to create a new Content Runtime and register any deployed instances to that Chef server again.

It is highly recommended that the Content Runtime virtual machine be backed up so that if it were ever destroyed, it could be restored without needing the following procedure.

Requirements

Get the following information:

- The Content Runtime host name
- The Content Runtime IP address
- The private and public keys from the original Content Runtime
- The network routing information from the original Content Runtime
- The deploy logs from any instances that had been deployed to the Content Runtime

AWS and IBM Cloud

To perform a recovery on any of these two cloud providers, a portable IP address had to be initially associated with the Content Runtime virtual machine. If that's the case, then the following information will be required in addition when creating the new Content Runtime:

AWS

- Private IP address

IBM Cloud

- Private subnet
- Private VLAN name
- Portable IP address
- Portable private netmask
- Portable private gateway

It'll also be required to backup the Pattern Manager's configuration file. This file is base64 encoded and can be found at **`/opt/ibm/docker/pattern-manager/config/config.json`**

Recovery actions

The following steps are required to recover your Content Runtime and register your existing deployed instances to the new Chef server on the Content Runtime:

1. **Deploy new Content Runtime**

   The first step in recovering your Chef server is to deploy a new Content Runtime instance. The new Content Runtime virtual machine must have the same host name and IP address as the old Content Runtime.

   Save the value used for the pattern manager access passphrase field. It is needed for making requests against the pattern manager.

   For more information on provisioning your Content Runtime, see Provisioning and managing your Content Runtime infrastructure.

2. **Populate software repositories**

   Repopulate your software repository as described in:

   - Populating the software repository
   - Populating the Installation Manager repository

3. **Optional: Restore your previous Chef server configuration**

   If a previous Chef server configuration was backed up from the previous Content Runtime virtual machine, it can be restored at this point by using the **`chef-server-ctl restore`** command. For more information visit Backup and Restore Chef.

4. **AWS and IBM only: Restore the Pattern Manager configuration file**

   Before copying the Pattern Manager's configuration back to the newly created Content Runtime virtual machine, there are some changes required.

   1. Open a terminal window on the location of the backed up **`config.json`**
   2. Run the following command replacing **`<previous_content_runtime_public_IP>`** with the previous Content Runtime's public IP address and **`<portable_or_private_IP>`** with the portable IP address (IBM) or the private IP address (AWS)

      ```
      base64 -d ./config.json | sed -e
      's/<previous_content_runtime_public_IP>/<portable_or_private_IP>/g' | base64 > config.json.new
      ```

      This command will decode the **`config.json`** file, replace the previous VM's IP address with the portable IP address (which will match with the newly create Content Runtime) and encoded the resulting change into a file named **`config.json.new`**
   3. Establish an SSH connection with the new Content Runtime virtual machine.
   4. Replace the contents of the current Pattern Manager configuration file, located at **`/opt/ibm/docker/pattern-manager/config/config.json`** with the ones from the generated **`config.json.new`**.
   5. Restart the Pattern Manager docker image by navigating to the **`advanced-content-runtime`** folder and running **`docker-compose stop`** followed by **`docker-compose start`**.

5. **Re-register your virtual machines**

   This step requires the Pattern Manager access passphrase that was specified when the Content Runtime was deployed.

   For each deployed instance:

   1. Log in to the Managed services web interface.
   2. Click **Menu** and click **Deployed Instances > Terraform templates**.
   3. Find your deployed instance in the list and click it.
   4. Click the **Log File** tab to bring up the log.

There are three REST requests made to the pattern manager in this log that you must reissue to re-register your virtual machine. The REST requests can be made from any environment that has network access to the Content Runtime. Some values have been removed from the log file and must be re-entered. The following examples use the **curl** command to make these requests, but they can also be made with any standard HTTP request tool.

1. **Create vault entry**

   The first request creates an empty vault entry for the stack ID of the previous deploy request. Search in the log file for a URL ending in **/v1/vault_item/chef**.

   Figure. Vault entry

   Copy the **camc_endpoint** and **data** properties and form a **curl** request. **Note**: If using IBM Cloud or AWS, instead of using the IP address provided in the **camc_endpoint**, use the "portable IP address" (IBM Cloud) or "private IP address" (AWS).

```
curl -i -H "Authorization: Bearer <pattern_manager_access_token>" -d '{"vault_content":
{"item":"secrets","values":{},"vault":"<vault-id>"}}'
https://<camc_endpoint_ip>:5443/v1/vault_item/chef
```

   Note: If there is no **VaultItem** in the deployment's logs, the vault id and **camc_endpoint_ip** can be obtained from the **bootstrap** section in the logs, as:

```
camc_bootstrap.LAMPNode01_chef_bootstrap_comp: Creating...
access_token:     "" => "******"
camc_endpoint:    "" => "https://<camc_endpoint_ip>:5443/v1/bootstrap/chef"
data:             "" => "
{\"environment_name\":\"_default\",\"host_ip\":\"host_ip_addr\",\"node_attributes\":
{\"ibm_internal\":{\"stack_id\":\"stack_id\",\"stack_name\":\"\",\"vault\":
{\"item\":\"secrets\",\"name\":\"<vault-
id>\"}}},\"node_name\":\"node_name\",\"os_admin_user\":\"root\",\"stack_id\":\"stack\"}"
```

   The request should return a 200 response code and the **vault_item** is added to the pattern manager.

2. **Bootstrap virtual machine**

   The next step is to bootstrap the virtual machine. A node on the Chef server for the virtual machine is created. Search in the log file for a URL ending in **/v1/boostrap/chef**.

   Figure. Bootstrap virtual machine

   Copy the **camc_endpoint** and **data** properties and form a **curl** request. **Note**: If using IBM Cloud or AWS, instead of using the IP address provided in the **camc_endpoint**, use the "portable IP address" (IBM Cloud) or "private IP address" (AWS).

```
curl -i -H "Authorization: Bearer <pattern_manager_access_token>" -d
'{"environment_name":"_default","host_ip":"9.5.38.18","node_attributes":{"ibm_internal":
{"stack_id":"77757fd10ef26711542b34f6ae5c368e","stack_name":"nschambu-mysql-01","vault":
{"item":"secrets","name":"77757fd10ef26711542b34f6ae5c368e"}}},"node_name":"wild-nschambu-
vm01","os_admin_user":"root","stack_id":"77757fd10ef26711542b34f6ae5c368e"}'
https://9.5.38.16:5443/v1/bootstrap/chef
```

   This request should return a 200 response code and a message confirming that the bootstrap request was successful.

3. **Assign software**

   The final step reassignes software to the endpoint. Search in the log file for a URL ending in **/v1/software_deployment/chef**.

Figure. Software deployment

Copy the **camc_endpoint** and **data** properties and form a **curl** request, adding a new property called **recovery_request** to the JSON data. **Note**: If using IBM Cloud or AWS, instead of using the IP address provided in the **camc_endpoint**, use the "portable IP address" (IBM Cloud) or "private IP address" (AWS).

```
curl -i -H "Authorization: Bearer <pattern_manager_access_token>" -d
'{"environment_name":"_default","host_ip":"9.5.38.18","node_attributes":{"ibm":
{"sw_repo":"https://9.5.38.16:9999","sw_repo_user":"repouser"},"ibm_internal":{"roles":"
[oracle_mysql_base]"},"mysql":{"config":{"data_dir":"/var/lib/mysql","databases":{"database_1":
{"database_name":"default_database","users":{"user_1":{"name":"defaultUser"},"user_2":
{"name":"defaultUser2"}}}},"log_file":"/var/log/mysqld.log","port":"3306"},"install_from_repo":"tru
e","os_users":{"daemon":
{"gid":"mysql","home":"/home/mysql","ldap_user":"false","name":"mysql","shell":"/bin/bash"}},"versi
on":"5.7.17"}},"node_name":"wild-nschambu-
vm01","os_admin_user":"root","runlist":"role[oracle_mysql_base]","stack_id":"77757fd10ef26711542b34
f6ae5c368e","vault_content":{"item":"secrets","values":{"ibm":{"sw_repo_password":"r3P0?
@5s"},"mysql":{"config":{"databases":{"database_1":{"users":{"user_1":
{"password":"aew2#4trNB"},"user_2":
{"password":"dwr3%%nGKMd"}}}}},"root_password":"fqt@9dbeft"}},"vault":"77757fd10ef26711542b34f6ae5c
368e"}, "recovery_request" : "True"}'
```

This request should return a 200 response code and a message confirming that your Chef runlist is updated to the previous cookbooks and roles.

Your deployed instances are now registered with the new Chef server. You are be able to interact with them in the Managed services web interface.

Troubleshooting

When executing the curl commands to re-register your virtual machine, a JSON object will be returned once the request has been executed. The following are examples of issues that might be found in this JSON message

```
HTTP/1.1 400 BAD REQUEST
Date: Mon, 11 Dec 2017 19:23:51 GMT
Server: Apache
x-pm-request-id: a5ba0df88f264cbaa940a0503a440b73
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Content-Length: 86
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: accept, authorization, content-type
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Type: application/json

{
  "message": "Vault e781c96d6a6b0f069b7e823877e045f3 item secrets already exists"
}
```

This error message indicates that the vault has already been created (step 1 on the **Re-register your virtual machines** has already been performed).

```
HTTP/1.1 400 BAD REQUEST
Date: Mon, 11 Dec 2017 19:19:58 GMT
Server: Apache
x-pm-request-id: a66f497afa8f4fe883eae340518ee2f5
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Content-Length: 80
```

```
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: accept, authorization, content-type
Access-Control-Allow-Methods: GET, POST, PUT, DELETE
X-Frame-Options: SAMEORIGIN
Connection: close
Content-Type: application/json

{
  "message": "vault_content was specified for a vault that does not exist."
}
```

The provided `vault-id` is either misspelled, or hasn't been created yet. Make sure that step 1 on the **Re-register your virtual machine** has been executed.

Bootstrap Fails With Knife Sudo Password Error

How to troubleshoot the bootstrap error when knife sudo fails

The IBM middleware cookbooks that are packages with Managed services (CAM) require a user with password-less sudo capabilities. For most templates, this user is root. However, if root is not available, then the operating system template that is used for deploy must include a user with the ability to use sudo without requiring a password.

If the deploy fails with this message, it means that the user did not have the ability to use sudo without requiring a password:

```
Error: Response from pattern manager:
StatusCode:500
Message:
{
 "message": "Bootstrap command failed. See the pattern manager logs for more details.",
 "rc": 1,
 "request_id": "1c844be316b74601853d2ddde595d94d",
 "stderr": "Creating new client for cam-apache-1\nCreating new node for cam-apache-1\nConnecting to
172.16.9.71\nstty: 'standard input': Inappropriate ioctl for device\nstty: 'standard input': Inappropriate
ioctl for device\nstty: 'standard input': Inappropriate ioctl for device",
 "stdout": "172.16.9.71 knife sudo password: \nEnter your password: \r\n172.16.9.71 \r\n172.16.9.71 Sorry,
try again.\r\n172.16.9.71 knife sudo password: \n172.16.9.71 \r\n172.16.9.71 Sorry, try again.\r\n172.16.9.71
knife sudo password: \n172.16.9.71 \r\n172.16.9.71 sudo: 3 incorrect password attempts"
}
```

Troubleshooting issues in IBM Cloud Pak for AIOps or Red Hat® OpenShift® Container Platform template deployments

Known issues and workaround related to IBM Cloud Pak for AIOps or Red Hat® OpenShift® Container Platform template deployment.

- When you destroy a failed IBM Cloud Pak for AIOps or Red Hat® OpenShift® Container Platform template deployment, your destroy operation may fail with the following error:

  ```
  Error applying plan:2 error(s) occurred:

    - module.gather_output.output.icp_install_dir: variable "get_cluster_config" is nil, but no error was
  reported
    - module.gather_output.output.registry_ca_cert: variable "get_cluster_config" is nil, but no error
  was reported
       Terraform does not automatically rollback in the face of errors.

       Instead, your Terraform state file has been partially updated with any resources that successfully
  completed.
       Please address the error above and apply again to incrementally change your infrastructure.
  ```

  This error is a warning that is thrown even if resources are destroyed. This is a known issue in Terraform v0.11.xx.

  For more information on this issue, see Link to issue reported by Terraform.

Troubleshooting Terraform related issues

Review the following known issues with Terraform.

- When you deploy a Terraform template that contains plug-in provider version that is not packaged with IBM Cloud Pak for AIOps, using Terraform engine 0.13.x or higher, you see the following error. The error is due to the changes made by Terraform on how Terraform manages the provider versions in Terraform 0.13.x or higher. For more information, see Terraform Provider Requirements.

  ```
  Failed to query available provider packages
  Could not retrieve the list of available versions for provider <provider_name>: no available releases
  match the given constraints <provider_version>
  ```

  To resolve this issue, you have to add the new provider version to the local provider registry. For more information, see Custom Terraform providers.

  For the list of shipped provider versions, see Shipped Terraform providers.

- After you install, you may see the following error when you deploy services.

  ```
  SubmitJobProviderResponseError: Failed to submit task for stack <stack_id> on the remote provider (API:
  /stack/create).  Caused by: The terraform binary for version <version_number> does not exist.
  ```

  The reason for this error is that the service definition in default provided services is using the older version of Terraform engine. The following versions of Terraform engine are packaged: 0.12.31, 0.13.7, and 0.14.11.

  To resolve this issue, edit the failed service in the service composer, open the source code tab and change the value for **provider_engine_version** attribute. If **provider_engine_version** is 0.12.21, then use 0.12.31.

Collecting data for Managed services

This section explains how and which data you must collect when you encounter problems in the Managed services. If you gather this information before contacting the IBM Support team, it expedites the troubleshooting process and saves time.

Self-diagnose

Before contacting IBM Support, you can try to find a resolution to the problem on your own. First, check Stack Overflow or the Troubleshooting sections to see whether your problem has been reported already.

If you are looking for examples, tutorials or more detailed information regarding specific Managed services topics, review the following links:

- Go through the other topics in this IBM Documentation.
- IBM Cloud Automation Manager Developer Community

If you see problems during the installation of Managed services, ensure that you have completed all the requirements correctly. That should solve most of your installation problems.

General troubleshooting information

If problem persists after self-diagnose, collect the following troubleshooting data:

- A detailed description of the problem. Be as precise as possible, include information where you see the issue, which steps you followed and if this can be reproduced every time or only happens intermittently.
- The error message (if any) and the timestamp of the error.
- Examples of the environment data that you can collect:
  - Production/test/staging environment
  - Managed services version
  - IBM Cloud Pak for AIOps version
  - Online or Offline installation
  - Topology
  - Proxy or network restrictions
  - Cloud provider
  - GlusterFS/NFS
  - Available disk space
- Screen captures that can help to understand the issue
- Did this work previously? If yes, what has changed? Any recent modifications to the system/configuration/version?
- Have any corrective actions been tried already?
- Did you apply any fixes from Fix Central or any other patches?
- Did you customize Kubernetes or any third-party component?
- Provide the business impact this issue is having on your company, including any upcoming deadlines as well as any realized or potential effect on customers.

Automatically collecting troubleshooting data

Use the **camCollector** script, which can be downloaded from GitHub or find it included in the Helm Chart.

For more information, see Collecting logs using camCollector script.

What to do next

After you collect the troubleshooting data, either review it on your own or send it to the IBM support team. If you have already opened a case for the issue, you can upload the data to that case directly by using the IBM Support Portal. If the data is too large or any other problems occur, see Enhanced Customer Data Repository to check alternative ways to exchange data.

For more complex issues, the recommendation is to open a case by using the IBM Support Portal.

Note: Do not send any confidential information from your company.

Refreshing internal certificates

Refresh internal certificates as follows if error with certificate expiry.

1. Switch to the cp4aiops namespace.

   ```
   oc project cp4aiops
   ```

2. Edit the certificate.

   ```
   oc edit certificate.certmanager.k8s.io cert
   ```

3. Add duration and renewBefore after **secretName: cert**.

   ```
   secretName: cert
   duration: 17520h
   renewBefore: 240h
   ```

4. After you save the changes, verify the output by using the command as follows:

   ```
   oc get certificate.certmanager.k8s.io cert -o yaml | grep -i duration -A4
   ```

   Output:

   ```
   duration: 17520h0m0s
   issuerRef:
     kind: Issuer
     name: cs-ca-issuer-for-cam
   renewBefore: 240h0m0s
   ```

5. Delete the secret certificate.

   ```
   oc delete secret cert
   ```

The Managed services (CAM) pods are restarted.

Collecting logs using camCollector script

The **camCollector** script is used to collect diagnostic information for Managed services and the underlying platform.

About the task

The script collects the following information:

- About Managed services Helm chart, such as the persistent volumes and its claims, secrets, config maps and deployments.
- OpenShift information for Managed services, such as namespaces/persistent volumes/pvc/secrets/config maps and status of each pod.
- Status of IBM Cloud Pak for AIOps pods.
- Logs for each of the Managed services pods.

Before you begin

To run the **camCollector** script, you must have configured OpenShift Client(oc).

Procedure

1. Ensure that you are logged in to your OpenShift Cluster with this command:

   ```
   oc login -n cp4aiops
   ```

   Note: If you use a namespace other than **cp4aiops**, you need to edit the **infrastructure_management_ns** variable in the script.

2. Download the **camCollector** script from GitHub using the following commands:

   ```
   curl https://raw.githubusercontent.com/IBM-CAMHub-Open/cam-support/release-2.0/scripts/camCollector.sh --output camCollector.sh
   ```

   ```
   chmod +x camCollector.sh
   ```

3. Run **camCollector.sh** script using the following command:

   ```
   ./camCollector.sh
   ```

Updating the resource limits for Managed services containers

To begin you need to install the correct version of the container in question, for example "ibm-management-cam-install.vx.xxxxxxx" and then specify the resource limit.

Steps involved in updating the resource limits for the container

1. Run the command:

```
oc project cp4aiops
```

2. Followed by:

```
oc edit iaconfig `oc get iaconfig --no-headers | awk '{print $1}'`
```

3. Next, update the content under the **ibm-management-cam-install** enablement section.

```
  - enabled: true
    name: ibm-management-cam-install
    spec:
      manageservice:
        camAPI:
          camMongo:
            resources:
              limits:
                cpu: 500m
                memory: 4Gi
              requests:
                cpu: 100m
                memory: 4Gi
```

4. Then, crosscheck the updates using the following command:

```
oc edit manageservice cam
```

5. The results should look like the following:

```
  camAPI:
    camMongo:
      resources:
        limits:
          cpu: 500m
          memory: 4Gi
        requests:
          cpu: 100m
          memory: 4Gi
```

6. Finally, run the command:

```
oc edit deploy cam-mongo
```

7. The results should match the resources you specified in the IAConfig and appear similar to the following:

```
containers:
        resources:
          limits:
            cpu: 500m
            memory: 4Gi
          requests:
            cpu: 100m
            memory: 4Gi
```

Support

If you need help with using IBM Cloud Pak® for AIOps, visit the IBM Support page.

Additionally, you can see whether the problem is documented in Known issues or in Troubleshooting.

Learn more about support from the following topics:

- Opening a support case
- Installing the MustGather tool
- Running the MustGather tool

Collecting support information for IBM Cloud Pak for AIOps

Learn how to gather information that is needed and relevant to a problem before you open a case with IBM® Support. Additionally, learn how to open a meaningful support case for IBM Cloud Pak® for AIOps.

Before you open a support case

Check the troubleshooting and known issues sections

Before you contact IBM Support, review the following topics to check whether your problem is already reported and a workaround or solution exists:

- Troubleshooting and support
- Known issues

Collect support information with MustGather tools

If you cannot find a solution for your problem in the documentation, complete the following steps to use the MustGather tool and gather information from the cluster for opening a support case. The collected data facilitates more effective support.

- Data collection for support cases
- Infrastructure Automation data collection for support cases

Data collection for support cases

1. Ensure that you meet the prerequisites for running the MustGather tool for collecting data
    - Access to the cluster as a user with a cluster-admin role.
    - The Red Hat® OpenShift® Container Platform CLI (oc) installed. For more information, see Installing the Red Hat OpenShift CLI.
2. Download and install the MustGather tool
3. Run the MustGather tool

Infrastructure Automation data collection for support cases

1. Ensure that you meet the prerequisites for running the MustGather tool for collecting data
    - Access to the cluster as a user with a cluster-admin role.
    - The Red Hat® OpenShift® Container Platform CLI (oc) installed. For more information, see Installing the Red Hat OpenShift CLI.
2. Run the MustGather to collect data and logs

   Follow the instructions on the following IBM Support page to run the MustGather tool to collect the data and logs for IBM Cloud Pak for AIOps: Collecting Data: MustGather for IBM Cloud Pak for AIOps - Infrastructure Automation

Opening a support case with IBM Support

If you cannot find a solution to your problem in product documentation, open a support case with IBM Support. Complete the following steps

1. Go to the IBM Support site and click **Open a case**.
2. Log in with your IBM ID and password, and consent to the IBM Privacy Statement.
3. Enter the case details. Enter the meaningful title that summarizes your problem. Then, select the Product Manufacturer, Product, and Severity.

   Note: The case severity is based on the business impact of the problem. If you set the case severity as 1, you need to have 24x7 availability to work with IBM support on this issue.
4. Provide the detailed description of your problem. A detailed description can help support understand your problem more accurately and thus provide solutions or answers more quickly. The following information is crucial:

   - Product version
   - Platform
   - Red Hat OpenShift Container Platform version
   - Virtualization platform (VMWARE, Azure, AWS, IBM Cloud)
   - High Availability (HA)
   - Steps to reproduce the issue
5. Optional: Upload the **`.tar.gz`** file with the results of the diagnostic scan. For more information, see Collecting support information.

   Note: If you encounter problems with uploading the file, see Uploading a file during case creation.
6. Select the language preferences and click **Submit a case**.

Installing the IBM Cloud Pak for AIOps MustGather tool

You must download and install the MustGather tool before you can run any commands for the tool. You must download the installation package for the MustGather tool from IBM Support. You can download the package to any machine that has valid access to the target cluster for running **`oc`** commands.

Important: This MustGather tool gathers data for IBM Cloud Pak® for AIOps deployments. For gathering data for Infrastructure Automation, see Infrastructure Automation data collection for support cases.

Procedure

Complete either of the following steps to download the tool:

- Download the (**`waiops-mustgather-<version>.tar.gz`**) package at the end of the following IBM Support page:

  Collecting Data: MustGather for IBM Cloud Pak for AIOps
- Open a command-line terminal and use the following command to download the package:

  ```
  wget https://www.ibm.com/support/pages/system/files/inline-files/waiops-mustgather-<version>.tar.gz
  ```

  Where **`<version>`** is the version of the MustGather tool. Refer to the preceding IBM Support page to determine the latest available version. For v.1.16.6, the command is:

  ```
  wget https://www.ibm.com/support/pages/system/files/inline-files/waiops-mustgather-v1.16.6.tar.gz
  ```

Important:

- Unpack the downloaded **`waiops-mustgather-<version>.tar.gz`** package in its own directory. Ensure that the directory is not shared with any other files or directories.
- Keep the MustGather tool up to date. By default the tool can run a **`new-version-check`** automatically. To disable this check, include the **`-M`** option when you run the MustGather tool to gather data. If you want to directly check for, and download, the latest version run the following command

  ```
  waiops-mustgather.sh -UD
  ```

MustGather tool use cases

Learn how to run the tool, and learn about the different use cases and command options that are available for use with the tool. For more information, see:

- Running the MustGather tool
- Data collection modes
- Advanced tips for using the MustGather tool

IBM Cloud Pak for AIOps MustGather tool

Learn how to use the IBM Cloud Pak® for AIOps MustGather tool to gather information that is needed and relevant to a problem before you open a case with IBM® Support.

Important: This MustGather tool gathers data for IBM Cloud Pak® for AIOps deployments. For gathering data for Infrastructure Automation, see Infrastructure Automation data collection for support cases.

The IBM Cloud Pak for AIOps MustGather tool is a wrapper for running `oc` commands. With this tool, you can complete data collection tasks and view analytics on the gathered data. After you run the tool to gather data, upload the output data file that is under your `/tmp` (or another directory if you used the ``-o` option) to the IBM Support website.

The tool generates a report that shows critical data about the status and health of the cluster and installation. This report allows the user view an overview of their cluster to help spot and solve multiple problems. The report can provide basic cluster analysis to identify cluster node resource utilization, warnings, and problematic resources. The report can also provide incident analysis, such as to identify whether Pod A is in CrashLoopBackOff.

The tool supports data collection for the following types of data:

Data type collection mode

Data type	Collection mode
`oc get` and `oc describe` (or `kubectl`) commands	This data is collected through the primary data collection modes.
Pod logs	This data is collected through the primary data collection modes.
Application logs within pods	This data collection is supported by the CPFILES secondary data collection mode.
Command output for commands that run through pods	This data collection is supported by the MANUALCOLLECT secondary data collection mode.
Command output that run on client machines (custom script)	This data collection is supported by the CMDEXEC secondary data collection mode.
Logs or command output that run on individual cluster nodes, such as journal, systemctl, iptables	This data collection is supported by the CLUSTERNODES secondary data collection mode.

Command syntax

```
waiops-mustgather.sh -[a|b|c|l] [-k resourname] [-z modules] [-d] [-L numlines] [-C cmdexec-file:cmdexec-env] [-E cmdexec-envvar] [-T timeout-value] [-m manual-collection-param] [-n manual-collection-envvar] [-j objname] [-w manual-collection-exec-timeout] [-e extra-namespaces:extra-resources] [-x product-version/path-to-config-file] [-A compliance-param] [-P plugins] [-Z plugins-envvar] [-Q plugins-timeout] [-F cpfiles-param] [-K] [-V view-only-module] [-y] [-J] [-p] [-t] [-g] [-f] [-R] [-S] [-s] [-u node-admin-username] [-i path-to-sshkey] [-o output-directory] [-G filename-regex:pattern] [-W primary-mode-var] [-N] [-I] [-X] [-Y] [-U] [-M] [-D] [-v] [-h]
```

- For details about the command options, see Data collection modes for the IBM Cloud Pak for AIOps MustGather tool.
- For details about running a custom script with the tool, see Run a custom script to complete corrective actions

Running the tool

- Primary use cases
- Additional use cases
- Miscellaneous use cases

Primary use cases

- Installation related problems
- Application related problems
- IBM Cloud Pak for AIOps related problems
- Log anomaly detection related problems

- Metric anomaly detection related problems
- Lifecycle related problems
- Datalayer related problems
- ChatOps related problems
- Healthcheck related problems

Installation related problems

To gather data for installation related problems, run the following command:

```
waiops-mustgather.sh -O install
```

The alias **-O install** is expanded to **-cypDfd** that uses the primary comprehensive (**-c**) mode and clusternode (**d**) mode to cause the tool to gather data for installation related problems.

To gather data for general non-installation related problems, run the following command:

```
waiops-mustgather.sh -O general
```

The alias **-O general** is expanded to **-cypDf** that is used to gather data for common problems.

Overall operation related problems

To gather data for overall IBM Cloud Pak for AIOps operation related problems, run the following command:

```
waiops-mustgather.sh -O aimgr
```

To gather raw data for IBM Cloud Pak for AIOps from its various data sources, such as Kafka, Flink, ElasticSearch and Postgres, run the following command:

```
waiops-mustgather.sh -O aimgr-data
```

To learn more about what aliases are available, use the command **waiops-mustgather.sh -O .listalias**

Important: The ALIAS mode (**-O xxx**) is used to expand an alias into a predefined command. If there is a need to change or add any option, you need to revert to the conventional way of running the MustGather tool by providing all the options or flags manually. For example, if a secret YAML is needed, you need to use **-cypDfS** (plus any other options or flags).

Log anomaly detection related problems

To gather data for log anomaly related problems, run the following command:

```
waiops-mustgather.sh -O lad
```

Metric anomaly detection related problems

To gather data for metric anomaly related problems, run the following command:

```
waiops-mustgather.sh -O mad
```

Lifecycle related problems

To gather data for Lifecycle related problems, run the following command:

```
waiops-mustgather.sh -O lifecycle
```

Datalayer related problems

To gather data for Datalayer related problems, run the following command:

```
waiops-mustgather.sh -O datalayer
```

ChatOps related problems

To gather data for ChatOps related problems, run the following command:

```
waiops-mustgather.sh -O chatops
```

Healthcheck related problems

To gather data for healthcheck related problems, run the following command:

```
waiops-mustgather.sh -O healthcheck
```

Additional use cases

- Run script to complete corrective actions

- Run baseline data collection
- Pod specific data collection
- Product specific data collection
- Installation verification
- Copying pod files and directories
- Complex command

Run a custom script to complete corrective actions

If you want to complete a **corrective action**, such as to delete a pod to restart it, you can write a script and run the following command to run the script:

```
waiops-mustgather.sh -C <script>
waiops-mustgather.sh -R -C <script>
```

Where **<script>** is the script that contains the corrective action to run.

The preceding command will only run your script. To perform data collection before running the script, you can opt for any of the primary modes."

Notes: If needed, you can run the secondary data collection modes **MANUALCOLLECT**, **EXTRA**, **CPFILES** within your custom script.

The following examples show how to trigger the MANUALCOLLECT, EXTRA, and CPFILES modes:

- To make sure the correction actions are successful, you can trigger MANUALCOLLECT to collect some data within the script

  ```
  echo "ALL##NS4PROD=aimanager##pod##teams######" >> $CMD_EXEC_RESULT_MCCFG_FILE
  ```

- You can also trigger EXTRA mode to collect namespace data within the script

  ```
  echo "openshift-insights" > $CMD_EXEC_RESULT_EXTRANS_FILE
  ```

- If you need to copy some files from some pods to make sure your script ran successfully:

  ```
  echo "NS4PROD=aimanager##^iaf-system-kafka####/opt/kafka/config/log4j.properties" >>
  $CMD_EXEC_RESULT_CPFILES_FILE
  ```

For an example custom script and some of the variables and functions that you can include within it, see Example: custom script.

Run baseline data collection

To run a baseline data collection, use any of the primary data collection modes, such as **-cypDf**, for example:

```
waiops-mustgather.sh -cypf -C <script> -e <namespace> -F 'NS4PROD=<prodname>,<podname-regex>,<container-
name>,<filename>'
```

With the preceding command, you can run a baseline data collection that is followed by running a corrective action through a custom script. You can also check the result of the corrective action by using the **-e** option where **<namespace>** is the namespace where the custom script runs. You can also opt to copy files from the restarted pod through the **-F** option.

Notes:

- Only **-c** is the primary data collection mode. Other options are supported, but should be used together.
- If you want to run another round of data collection after your custom script runs, use a secondary data collection mode (other than **clusternodes** or **cmdexec**).

Pod specific data collection

If you have to run commands on a specific pod, but you do not know the full name of the pod, such when you know only the UUID portion, you can use the **manualcollect** mode:

```
waiops-mustgather.sh -R -m ps-out:[configfile]
```

Example (**<configfile>**):

```
@ TAG##NAMESPACE##OBJTYPE##OBJNAME##CONTAINER##EXECPREP##EXECCMD
ps-out##NS4PROD=aimanager##pod##^cp4waiops-postgres-keeper######hostname;ps -ef
```

The podname is a regex. The tool determines the full name of the pod automatically.

```
[OUTPUT - found in file 4-MANUALCOLLECT/<namespace>/<podname>.bash.exec]
cp4waiops-postgres-keeper-0
UID          PID    PPID  C STIME TTY          TIME CMD
stolon         1       0  1 Nov09 ?        02:09:11 stolon-keeper --data-dir /stolon-data
stolon        29       1  0 Nov09 ?        00:00:00 [create-template] <defunct>
stolon        54       1  0 Nov09 ?        00:15:06 postgres -D /stolon-data/postgres -c
unix_socket_directories=/tmp
stolon        59      54  0 Nov09 ?        00:00:09 postgres: checkpointer
stolon        60      54  0 Nov09 ?        00:00:10 postgres: background writer
```

```
stolon        61      54  0 Nov09 ?        00:01:00 postgres: walwriter
stolon        62      54  0 Nov09 ?        00:00:19 postgres: autovacuum launcher
stolon        63      54  0 Nov09 ?        00:02:47 postgres: stats collector
stolon        64      54  0 Nov09 ?        00:00:00 postgres: logical replication launcher
stolon   2905619      54  0 05:39 ?        00:00:00 postgres: cp4aiops cp4aiops 10.254.12.46(50562) idle
stolon   2906589      54  0 05:43 ?        00:00:00 postgres: cp4aiops cp4aiops 10.254.12.46(53384) idle
stolon   2918996       0  0 06:34 ?        00:00:00 bash -c hostname;ps -ef
stolon   2919006 2918996  0 06:34 ?        00:00:00 ps -ef
```

Product specific data collection

If you need to collect product-specific data, you can run the *plugins* secondary collection mode:

```
waiops-mustgather.sh -P aimanager
```

Installation verification

If you just want to check whether an installation is successful:

```
waiops-mustgather.sh -x X -DR
```

The preceding command scans the product namespaces for missing resources or objects based on the configuration file that is specified in the **<MUSTGATHER-DIR>/missingobj/config/missingobj-waiops-<prodver>.cfg** file.

Copying pod files and directories

If you need to copy files from a pod, you can run a command similar to the following command:

```
waiops-mustgather.sh -F 'NS4PROD=aimanager##aimanager-aio-log-anomaly-detector-####/opt/ai4it/*.log*'
```

The preceding command copies files and directories that match the filename wildcard of **/opt/ai4it/*.log*** from pods that match regex **aimanager-aio-log-anomaly-detector-** in the auto-resolved IBM Cloud Pak for AIOps namespace.

Complex command

If needed, you can run a more complex command:

```
waiops-mustgather.sh -aypf -e ibm-common-services,openshift-operators -P aimanager -G
'ALL:error|fail|exception|login' -C /tmp/test.sh -E 'TEST1=uname -a##TEST2=date' -m cqlsh:manualcollect.csv -
n 'SQL="describe tables"' -F spark
```

Miscellaneous use cases

- Scale down the MustGather data collection

Scale down the MustGather data collection

If the cluster that you are working on is slow or throttles a lot, you can scale down the MustGather through the *selective* primary collection mode:

```
waiops-mustgather.sh -Xy -k deployments,sts,pods,jobs,pvc
```

The preceding command disables custom data collection (**-X**), enables YAML collection (**-y**) and runs the MustGather tool with the *selective* primary collection mode (**-k**) to collect data on resources such as **deployments**, **statefulsets**, **pods**, **jobs**, and **pvc**.

Command help

If you need help with running the MustGather tool, you can use the tool **help** command when you run the tool to view more details about the command:

```
waiops-mustgather.sh -h
```

Data collection modes for the IBM Cloud Pak for AIOps MustGather tool

Learn more about the different commands that you can use with the IBM Cloud Pak® for AIOps MustGather tool to gather information for opening a case with IBM® Support.

There are 2 sets of data collection modes (primary and secondary). The MustGather tool has multiple (6) primary collection modes and multiple advanced modes for collecting specific sets of data.

Rules:

- Users can select zero or one primary mode.

- User can select zero, some or all secondary modes.
- User can combine one primary mode with zero, some or all secondary modes.
- Secondary modes can run independently without any primary mode
- The secondary modes selected (independently, or with any primary mode) are run with the following precedence: **clusternodes** > **cmdexec** > **manualcollect** > **extra** > **missingobj** > **compliance** > **plugins** > **cpfiles** > **snapshots**

Example commands:

```
waiops-mustgather.sh -aypD -C /tmp/cmdexec.input -m /tmp/manualcollect.csv -e ibm-common-services
```

Since a primary mode **-a** is selected, all data collection that is configured for the *all* primary mode is run first. Then the data collection for the secondary modes are run in the sequence of **cmdexec** > **manualcollect** > **extra**.

```
waiops-mustgather.sh -C /tmp/cmdexec.input -m /tmp/manualcollect.csv -e ibm-common-services
```

Since no primary mode is selected, the secondary modes are run in the sequence **cmdexec** > **manualcollect** > **extra**.

Primary modes

Primary mode options

Option	Description
-a	All mode, which gets all data (data volume is high).
-c	Comprehensive mode, which gets comprehensive data (data volume is moderate).
-b	Basic mode, which gets basic data (data volume is OK).
-l	Limited mode, which gets limited data (data volume is low).
-k	Selective mode, which gets data for selective resources (separate resources with comma), for example, pods,deployment,secrets (data volume is varied). To get cluster-wide (all-namespaces) data for namespace-scoped resource, such as a pod, you can use the special keyword **allns** (**pod/allns**,**ingress**,**cm/allns**). **allns** data is available in the **CLUSTER_DATA** directory. **namespaced** data is available in the **PROD_NAMESPACES** directory.
-z	Module mode, which gets data based on predefined modules separated by comma (clusterinfo productinfo storage networking installation csv pvcusage nodeinfo) (data volume is varied). To list all available modules, use **-z .listmodules**.

Secondary modes

Clusternode modes

Clusternode mode options

Option	Description
-d	Gets logs and data from all cluster nodes by using **oc debug** or SSH (if **-i** or **-u** is used).
-l	Number of lines of journal to collect from cluster nodes. The default is **2000**.

cmdexec mode

cmdexec mode options

Option	Description
-C	**[script file]##[environment file]** Where **[environment file]** is optional. The format of **[environment file]** is **KEY=VALUE**, which is similar to the variable definition format in shell script.
-E	Environment variable for the **cmdexec** script. For example, **'VAR1=VAL1##VAR2=VAL2##...'** - Enclose the entire input string with single quotation marks. For example **'VAR1=hello##VAR2=uname -a'**
-T	Timeout value for **cmdexec [-C]**. The default is **5m**. For example, **10s** where **s**=second; **m**=minute; **h**=hour.

Manualcollect mode

Manualcollect mode options

Option	Description
-m	**<manual-collection-param>** = **[TAGS]:input file** for the **getManualCollection()** function where **TAGS** is optional. The default is all config lines. - Separate multiple tags with a comma.
-n	Environment variables for the **manualcollect** **EXECCMD** command. For example, **TAG1~>VAR1=VAL1##VAR2=VAL2^^TAG2~>VAR1=VAL1...** - Use the symbol **^^** to separate environment variables for different tags. - Use the symbol **~>** to define an environment variable for a tag. Use the symbol **##** to separate multiple environment variables. - Enclose the entire input string with single quotation marks and each value with double quotation marks. For example, **VAR1="hello"##VAR2="uname -a"**.

Option	Description
`-j`	Provided string `[tag:<objname>##tag:<objname>##...]` is used to substitute the `OBJNAME` keyword under the `OBJNAME` column that is based on the `tagname` in the manual collect input configuration file, such as `tag1=pod-abc##tag2=pod-xyz`.
`-w`	Timeout value for `manualcollect [-m]`. The default is `5m`. For example, `10s` where `s`=second; `m`=minute; `h`=hour.

Extra namespace mode

Extra namespace mode options

Option	Description
`-e`	Get data for extra namespaces (`oc get all,pvc,configmap,secret,serviceaccount`, `oc describe`, `oc logs`) - Separate namespaces with a comma (`,`)

The `[:extra-resouces]` is optional and is a file that can be used to specify additional resource types, such as catalog sources, operand configs, and more). Specify one resource type per line in the file.

Missing object mode

Missing object mode options

Option	Description
`-x`	- Product version to use for `getMissingObj` function, such `-x 3.2.0` or `-x X` to auto-detect. Defaults to the product version that is detected by tool if a version is not provided Or - The absolute path for an input config file, such as `-x /tmp/myfile.cfg` for the `getMissingObj` function to use.

Compliance mode

Compliance mode options

Option	Description
`-A`	`<compliance-param> = TAGS:[input file for the getCompliance() function]` where `input file` is defaulted to `compliance/<prod-version>/compliance.cfg` (use keyword `ALL` to select all config lines. Separate multiple tags with a comma) [EXPERIMENTAL]

plugins mode

Plugins mode options

Option	Description
`-P`	Product plugin `[supported=aimanager]:action [supported=data fix]`. The default action is `data`. For example, `aimanager` or `aimanager:data` or `aimanager:fix`. - Separate plugins with a comma (`,`).
`-Z`	Environment variables for the plugins script. For example, `VAR1=VAL1##VAR2=VAL2##...` - Enclose the entire input string with single quotation marks. For example, `VAR1=hello##VAR2=uname -a`.
`-Q`	Timeout value for plugins `[-P]` The default is `15m`. For example, `10s` where `s`=second; `m`=minute; `h`=hour.

cpfiles mode

cpfiles mode options

Option	Description
`-F`	Copy files in a pod. `cpfiles-param` = `TAGS` or `NAMESPACE##PODNAME##CONTAINER##FILES` where multiple `TAGS` can be delimited by `@@`. `CONTAINER` is optional and `FILES` can be delimited by a colon. - `-F aimanager` refers to `cpfiles/cpfiles-<prodname>-<version>.cfg` and copies all the files configured as `TAG=aimanager`. `-F NS4PROD=aimanager##iaf-system-kafka-0####/tmp/abc.txt` is the manual command to copy a file.

Notes:

- You can use keyword `NS4PROD=<prodname>` to obtain namespace for those products automatically.
- You can use asterisk(`*`), square brackets(`[]`), question mark(`?`), and curly braces(`{}`) wildcards in `FILES`.
- Since the `tar` (needed by `oc cp`) command might not be available in the target pod or container, use this option only for text files (`oc exec/cat` is used as a secondary option).

Snapshots mode

Snapshots mode options

Option	Description
`-K`	Complete snapshots comparison.

View mode

View mode options

Option	Description
`-V`	`<view-only-module>` is a script that can be created under the `<mustgather-install-dir>/config/view/<PRODUCT_VERSION>` (precedence) or `<mustgather-install-dir>/view` directory. You can use `-V .listviews` to list down all available modules and their descriptions.

Supporting Options

Supporting options

Option	Description
`-y`	Collect YAML output (`oc get -o yaml`).
`-J`	Collect JSON output (`oc get -o json`).
`-p`	Collect logs for the previous instance of the container in a pod if it exists
`-t`	Do not collect `oc describe`.
`-g`	Do not collect pod logs (`oc logs` and `oc logs -p`).
`-f`	Collect PVC usage data (`df -h`) - Available only with options `-a`, `-c`, `-b`, `-l`, `-z` (productinfo/storage/pvcusage), `-k` (pvc) and `-e`.
`-R`	Enable report-on-screen where SUMMARY.log is printed on display.
`-S`	Explicitly collect YAML output of secret objects.
`-s`	Use SSH to collect logs and data from all cluster nodes without a common admin username across the cluster.
`-u`	Provide the admin username to used to SSH to all cluster nodes for data collection purpose. A passwordless SSH is needed for this option.
`-i`	Set the path to the SSH key. This path is needed for clusters on a cloud (experimental).
`-o`	Set the output directory. The default directory is `/tmp`.
`-G`	Use a filename regex and pattern, that is delimited by a colon, to run a grep operation. This option is applicable to pod logs, events, cpfiles, and node-related data files. - Separate patterns with double asperands (`@@`). - Separate `FILENAME_REGEX:PATTERNS` key value pairs with double hashes (`##`). - Use the keyword `ALL` as the filename regex as an instruction to run `grep` for all detected files. - Use a tilde (`~`) to run a piping grep. For example, `grep A <file>` | `grep B` | `grep C`. - To escape, use a tilde (`~`) with a backward slash (\) for the literal value. There is no need to escape by using a colon (`:`). For example, `ALL:error@@failure` or `^aimanager:error~fail` or `^aimanager:error##ALL:failure` or `ALL:error\~fail##ALL:error\/fail`
`-W`	Set global environment variable that all primary data collection modes can use for further processing. For example, `-W 'VAR1=123##VAR2=456##VAR3=789'`; this option delimits multiple variables with the `##` characters. All variables are suffixed with the string `PRIMODE_` when used internally (`VAR1` > `PRIMODE_VAR1`).
`-N`	Turn on data collection for problematic non-product namespaces. Data collection for problematic Red Hat OpenShift namespaces is always enabled.
`-I`	Enable incident reporting (experimental).
`-X`	Turn off custom data collection.
`-Y`	Auto-answer `YES` to question prompts.
`-U`	Check for updates.
`-M`	Turn of the automatic new version check and download for the tool.
`-D`	Turn on Debug mode.
`-v`	View the version details.
`-h`	View the tool help details.

Advanced tips for using the IBM Cloud Pak for AIOps MustGather tool

Learn more about advanced techniques that you can use with the IBM Cloud Pak® for AIOps MustGather tool to gather information for opening a case with IBM® Support.

Tip: Use manual collection mode to run commands

If you need to run `oc exec` commands on pods, you can use the manual collection (`MANUALCOLLECT`) mode. This mode can be configured to complete specific tasks, such as to "get all Elasticsearch indexes".

The MustGather Tool does include some default task configurations. For more information, review the **manualcollect/<version>/mc.cfg** file within the MustGather tool installation package.

Example command:

```
# ./waiops-mustgather.sh -R -m aimgr-indices:mc.cfg
```

Example output:

```
================================================
[MANUALCOLLECT] OUTPUT OF EXECCMD
================================================

[CFGLINE = lifecycle-policy##cp4waiops##pod##^aiops-ir-lifecycle-policy-registry-svc-######curl -u$(cat
$SYSTEMAUTH_BINDING_DIR/username):$(cat $SYSTEMAUTH_BINDING_DIR/password)
http://localhost:$PORT/policyregistry.ibm-netcool-prod.aiops.io/v1alpha/system/cfd95b7e-3bc7-4006-a4a8-
a73a79c71255]

[FILE = /tmp/waiops-manualcollect-07112022-190154/4-MANUALCOLLECT/TAG=lifecycle-policy/cp4waiops/aiops-ir-
lifecycle-policy-registry-svc-7587b88cdb-rk594.bash.exec]

[POD = aiops-ir-lifecycle-policy-registry-svc-7587b88cdb-rk594]

Defaulted container "aiops-ir-lifecycle-policy-registry-svc" out of: aiops-ir-lifecycle-policy-registry-svc,
policyregistry-checkdb (init)

[

  {

    "tenantid": "cfd95b7e-3bc7-4006-a4a8-a73a79c71255",

    "policyid": "80301820-406a-11ed-bf66-4dfc9e76db68",

    "configuration": "{\"executionPriority\":80,\"state\":\"enabled\",\"spec\":{\"triggers\":
[{\"entityId\":\"alert\",\"triggerId\":\"aiops.ibm.com/trigger/alert-pre-update\",\"arguments\":
{\"condition\":\"{{#if (exists alert.sender.name)}} {{ alert.sender.name }} == \\\"Log Anomaly\\\" and not {{
isEmpty alert.insights }} {{else}} false {{/if}}\"}},
{\"entityId\":\"alert\",\"triggerId\":\"aiops.ibm.com/trigger/alert-pre-create\",\"arguments\":
{\"condition\":\"{{#if (exists alert.sender.name)}} {{ alert.sender.name }} == \\\"Log Anomaly\\\" and not {{
isEmpty alert.insights }} {{else}} false {{/if}}\"}}],\"actions\":
[{\"actionId\":\"aiops.ibm.com/action/internal/partition\",\"arguments\":{\"partitionKey\":
{\"$variable\":\"alert.id\"},\"actions\":
[{\"actionId\":\"aiops.ibm.com/action/internal/array/foreach\",\"arguments\":{\"array\":
{\"$variable\":\"alert.insights\"},\"condition\":\"{{element.type}} == \\\"aiops.ibm.com/insight-
type/lad/templates\\\"\",\"foreach\":
[{\"actionId\":\"aiops.ibm.com/action/internal/insights/aggregate\",\"arguments\":{\"aggregationKey\":
{\"$variable\":\"alert.id\"},\"insightDetails\":{\"$variable\":\"element.details\"},\"insightType\":
{\"$variable\":\"element.type\"},\"insightId\":{\"$variable\":\"element.id\"}},\"output\":\"alert.insights.
[@id=insight-
lad]\"}]}}]}}]},\"hash\":\"7d1f1b4fbb433259aaf0bf2ffc3305de06e41d98\",\"revision\":\"ee305a657825c73944447ec2
ffdedd6d6519530d\",\"hotfields\":[]}",

    "entityid": "[\"alert\",\"alert\"]",

    "metadata": "{\"labels\":{\"ibm.com/is-default\":\"true\"},\"name\":\"Aggregate log anomaly detection
count vectors\",\"description\":\"Aggregates the log message count vectors output by log anomaly events,
across the event occurences contributing to a given alert.\\n\",\"createdBy\":
{\"id\":\"system\",\"type\":\"system\"},\"lastUpdatedBy\":
{\"id\":\"system\",\"type\":\"system\",\"changeDetails\":\"Created\"},\"lastUpdatedTimestamp\":\"2022-09-
30T02:49:27.969Z\",\"creationTimestamp\":\"2022-09-30T02:49:27.969Z\"}",

    "triggerid": "[\"aiops.ibm.com/trigger/alert-pre-update\",\"aiops.ibm.com/trigger/alert-pre-create\"]"
  },
...
```

Tip: Use `CPFILES` to copy files from pods

If you need to copy specific files from pods, you can use the **CPFILES** option.

For example, the following command uses the MustGather tool to pull the **/conf/redis/redis.conf** file from all pods that match regex expression **^c-example-redis-m-**. This example uses the following parameters:

- **NS4PROD=** is a special keyword that instructs the tool to auto-resolve the namespace for product **aimanager**.
- **keyword** and **regex** (avoid hardcoding) are used to allow the configuration to be used across different environments without the need to reconfigure.

Example command:

```
[file cpfiles/cpfiles-waiops-320.cfg in the package]
test##NS4PROD=aimanager##^c-example-redis-m-####/conf/redis/redis.conf
```

Example output:

```
# pwd
/tmp/waiops-cpfiles-01042022-173635/8-CPFILES/waiops32
# ls -l c-example-redis-m-0/
total 56
-rw-r--r-- 1 root root 51281 Apr  1 17:36 '^conf^redis^redis.conf'
-rw-r--r-- 1 root root    78 Apr  1 17:36 '^conf^redis^redis.conf.ls'
# ls -l c-example-redis-m-1
total 56
-rw-r--r-- 1 root root 51306 Apr  1 17:36 '^conf^redis^redis.conf'
-rw-r--r-- 1 root root    78 Apr  1 17:36 '^conf^redis^redis.conf.ls'
# ls -l c-example-redis-m-2
total 56
-rw-r--r-- 1 root root 51306 Apr  1 17:36 '^conf^redis^redis.conf'
-rw-r--r-- 1 root root    78 Apr  1 17:36 '^conf^redis^redis.conf.ls'
```

Tip: Run custom scripts with the MustGather tool

You can run your own script with the MustGather Tool, such as for completing corrective actions before the tool collects data. After completing the custom script actions, the tool then proceeds with the data collection.

- Custom script variable
- Custom script functions
- Example Command
- Example script
- Example script functions
- Example script results

Custom script variable

The following tables list some of the variables that you can include within a custom script:

Custom script variables

Variable	Description
`$OC_CMD`	The system command used to collect data. You can use either an `oc` or `kubectl` command.
`$OC_WHOAMI`	The current user.
`$AWK`	The `AWK` used in the tool, which can be either `awk` or `gawk` (preferred).
`$OLDIFS`	The original IFS.
`$PRODUCT_VERSION`	The product version detected during runtime (on site).
`$MUSTGATHER_DIRNAME`	The directory name where the MustGather tool is installed.
`$OUTDIR`	The directory where the output data is written.
`$PROD_NAMESPACES_REFFILE`	The product-to-namespace mapping reference file (Do not change).
`$PROD_NAMESPACES_UNIQUE_REFFILE`	The unique product namespaces reference file (Do not change).
`$PROD_NAMESPACES_INVALID_STR`	A constant string value that represents the failed `getNamespaceToProdMapping()` function.
`$CMD_EXEC_OUTPUT_DIR`	The directory where the output of `CMDEXEC` is written to.
`$CMD_EXEC_PRODUCT_NSFILE`	A copy of the original product namespaces reference file that needs to be used by the inherited `getNamespaceToProdMapping()` function.
`$CMD_EXEC_RESULT_MCCFG_FILE`	A file that the custom script can write to for running the *MANUALCOLLECT* secondary mode.
`$CMD_EXEC_RESULT_EXTRANS_FILE`	A file that the custom script can write to for running the *EXTRA* (namespace) secondary mode.
`$CMD_EXEC_RESULT_CPFILES_FILE`	A file that the custom script can write to for running the *CPFILES* secondary mode.

Custom script functions

The following tables list some of the functions that you can include within a custom script:

Custom script functions

Functions	Description

Functions	Description
`getProdToNamespacesMapping()`	A function to retrieve product namespace through provided product name. **Syntax:** `getProdToNamespacesMapping <product-name> $CMD_EXEC_PRODUCT_NSFILE` **Returns:** - Product namespaces of the given product name or `PROD_NAMESPACES_INVALID_STR`. - Return status `0` for successful execution and `1` for a failed execution. Product namespaces: - `aimanager` = IBM Cloud Pak for AIOps - `noihybrid` = IBM Netcool Operations Insight (hybrid) - `ics` = IBM Cloud Pak foundational services
`getNamespaceToProdMapping()`	A function to retrieve product name through provided product namespace. **Syntax:** `getNamespaceToProdMapping <product-namespace> $CMD_EXEC_PRODUCT_NSFILE` **Returns:** Product name of the given product namespace.
`getResourceInstances()`	A function to retrieve a list of resource instances. **Syntax:** `getResourceInstances <resource type> <resource name regex> <resource namespace>` Where `<resource namespace>` is optional and defaults to `-all-namespaces`. **Returns:** - A list of resources that match the provided details or a blank string if the command fails or no resource is detected. - Return status `0` for successful execution and `1` for a failed execution.
`getUniqueProdNamespaces()`	A function to retrieve a list of unique product namespaces. **Syntax:** `getUniqueProdNamespaces $CMD_EXEC_PRODUCT_NSFILE` **Returns:** A list of unique product namespaces or blank string if failed.
`checkUserPriv()`	A function to check whether the current user has privilege to complete specific actions (verb) on a specific resource in a specific namespace. **Syntax** `checkUserPriv <verb> <resource type> <resource namespace>` **Returns:** `0` to indicate the user has the required privileges and `1` to indicate that user does not have the required privilege. `<verb>` reference: Kubernetes API Concepts.
`isPodOk()`	A function to determine whether a pod is in `RUNNING` state and all its containers are up. **Syntax:** `isPodOk <namespace> <podname>` **Returns:** `0` for an OK pod and `1` for a pod that is not OK.
`isPodLineOk()`	A function to determine whether a pod is in RUNNING state and all its containers are up. **Syntax:** `isPodLineOk <'oc get –no-headers' output of the pod>` **Returns:** `0` for an OK pod and `1` for an pod that is not OK.

Example Command

```
waiops-mustgather.sh -R -C /tmp/elite.sh##/tmp/env.in -Y
```

The previous command uses an environment variable file `/tmp/env.in` to pass in variables into the script.

```
POD=teams
CMD=date > /tmp/date.out
```

Example script

The following code shows an example custom script:

```
NOI_NAMESPACE=$(getProdToNamespacesMapping noi $CMD_EXEC_PRODUCT_NSFILE)

if [ ${#NOI_NAMESPACE} -gt 0 ]
then
  echo "[$(date)] Detected namespace = [$NOI_NAMESPACE]"

  IMPACTGUI_POD=$(getResourceInstances pod impactgui $NOI_NAMESPACE)

  if [ ${#IMPACTGUI_POD} -gt 0 ]
  then
    echo "[$(date)] Detected IMPACT GUI pod = [$IMPACTGUI_POD]"
    oc delete pod -n $NOI_NAMESPACE $IMPACTGUI_POD

    WAITFORPOD=0

    while [ $WAITFORPOD -eq 0 ]
    do
      if isPodOk "$NOI_NAMESPACE" "$IMPACTGUI_POD"
      then
        OUTFILE=$($OC_CMD exec -n $NOI_NAMESPACE $IMPACTGUI_POD --
/opt/IBM/tivoli/impact/bin/nci_collect_logs | grep 'Netcool/Impact log file is' | $AWK '{ print $7 }')
```

```
        if [ ${#OUTFILE} -gt 0 ]
        then
          echo "[$(date)] Creating COLLECTLOG timestamp file..."
          echo "ALL##$NOI_NAMESPACE##pod##$(getResourceInstances pod impactgui $NOI_NAMESPACE)####touch
/tmp/collectlog" > $CMD_EXEC_RESULT_MCCFG_FILE

          echo "[$(date)] Detected IMPACT GUI logfile = [$OUTFILE]"
          echo "$NOI_NAMESPACE##$IMPACTGUI_POD####$OUTFILE" > $CMD_EXEC_RESULT_CPFILES_FILE

          WAITFORPOD=1
        else
          echo "[$(date)] Unable to locate IMPACT GUI logfiles!"
          exit 1
        fi
      else
        echo "[$(date)] IMPACT GUI pod is not running!"
        sleep 5
      fi
    done
  else
    echo "[$(date)] Unable to locate IMPACT GUI pod!"
    exit 1
  fi
else
  echo "[$(date)] Unable to determine NOI namespace!"
  exit 1
fi
```

Example script functions

When you are creating your custom script, you can use the following functions:

- The function **getProdToNamespacesMapping** can be used in any **CMDEXEC** script to obtain product namespaces:

  ```
  NS=$(getProdToNamespacesMapping aimanager $CMD_EXEC_PRODUCT_NSFILE)
  echo "NAMESPACE=$NS"
  ```

- The Function **getResourceInstances** can be used in any **CMDEXEC** script to obtain resource instances based on a provided regex expression such to complete a corrective action:

  ```
  for MYPOD in $(getResourceInstances pod $POD $NS)
  do
  MYCMD="oc exec $MYPOD -n $NS -- bash -c '$CMD'"
  echo $MYCMD
  eval $MYCMD
  done
  ```

- The **MANUALCOLLECT** mode can be used to collect data within the script, such as to ensure that the correction actions are successful:

  ```
  echo "ALL##NS4PROD=aimanager##pod##teams####" >> $CMD_EXEC_RESULT_MCCFG_FILE
  ```

- The **EXTRA** mode can be used to collect namespace data within the script:

  ```
  echo "openshift-insights" > $CMD_EXEC_RESULT_EXTRANS_FILE
  ```

- The **CPFILES** option can be used to copy files from pods to verify that your script ran successfully:

  ```
  echo "NS4PROD=aimanager##^iaf-system-kafka####/opt/kafka/config/log4j.properties" >>
  $CMD_EXEC_RESULT_CPFILES_FILE
  ```

Example script results

- When the command **date > /tmp/date.out** runs in the pod:

  ```
  oc exec -it
  ```

  Output:

  ```
  aimanager-aio-chatops-teams-integrator-545bc7d4d-9w8fp -- bash -c 'ls -l /tmp; cat /tmp/date.out'
  Defaulted container "chatops-teams-integrator" out of: chatops-teams-integrator, controller-is-ready
  (init)
  total 12
  -rw-r--r--. 1 default root  29 Apr  2 01:00 date.out
  -rwx------. 1 root    root 701 Sep 14  2021 ks-script-vzv5jj6c
  -rwx------. 1 root    root 291 Sep 14  2021 ks-script-y5xeen0d
  Sat Apr  2 01:00:25 UTC 2022
  ```

- Through **MANUALCOLLECT**, the data of the pod that matched the regex **teams** is collected. You can verify this by running the following commands:

1. Print the working directory:

   ```
   # pwd
   ```

   Output:

   ```
   /tmp/waiops-cmdexec-01042022-175950/4-MANUALCOLLECT/TAG=ALL/waiops32
   ```

2. List the directory files:

   ```
   # ls -l
   ```

   Output:

   ```
   total 4
   -rw-r--r-- 1 root root 180 Apr  1 18:00 oc_pod-aimanager-aio-chatops-teams-integrator-545bc7d4d-
   9w8fp.out
   drwxr-xr-x 2 root root 164 Apr  1 18:00 pod
   ```

3. List the pods files:

   ```
   total 12
   -rw-r--r-- 1 root root 1551 Apr  1 18:00 aimanager-aio-chatops-teams-integrator-545bc7d4d-
   9w8fp~chatops-teams-integrator.log
   -rw-r--r-- 1 root root 7315 Apr  1 18:00 aimanager-aio-chatops-teams-integrator-545bc7d4d-
   9w8fp.desc
   ```

- Through the **EXTRA** mode, data about the namespace **openshift-insights** is collected. You can verify this by running the following commands:

  1. Print the working directory:

     ```
     # pwd
     ```

     Output:

     ```
     /tmp/waiops-cmdexec-01042022-175950/5-EXTRA/openshift-insights
     ```

  2. List the directory files:

     ```
     # ls -l
     ```

     Output:

     ```
     total 28
     drwxr-xr-x 2 root root  132 Apr  1 18:00 configmaps
     drwxr-xr-x 2 root root   36 Apr  1 18:00 deployment.apps
     -rw-r--r-- 1 root root 1738 Apr  1 18:00 oc_all.out
     -rw-r--r-- 1 root root  194 Apr  1 18:00 oc_configmaps.out
     -rw-r--r-- 1 root root  328 Apr  1 18:00 oc_pod_scc.out
     -rw-r--r-- 1 root root   52 Apr  1 18:00 oc_pvc.out
     -rw-r--r-- 1 root root 1427 Apr  1 18:00 oc_secrets.out
     -rw-r--r-- 1 root root  155 Apr  1 18:00 oc_serviceaccounts.out
     drwxr-xr-x 2 root root  117 Apr  1 18:00 pod
     drwxr-xr-x 2 root root   35 Apr  1 18:00 pvc
     drwxr-xr-x 2 root root   86 Apr  1 18:00 replicaset.apps
     drwxr-xr-x 2 root root 4096 Apr  1 18:00 secrets
     drwxr-xr-x 2 root root   26 Apr  1 18:00 service
     drwxr-xr-x 2 root root  107 Apr  1 18:00 serviceaccounts
     ```

- Through CPFILES, the file **/opt/kafka/config/log4j.properties** is copied from the pods that matches the regex expression **^iaf-system-kafka**. You can verify this by running the following commands:

  1. Print the working directory:

     ```
     # pwd
     ```

     Output:

     ```
     /tmp/waiops-cmdexec-01042022-175950/8-CPFILES/waiops32/iaf-system-kafka-0
     ```

  2. List the directory files:

     ```
     # ls -l
     ```

     Output:

     ```
     total 12
     -rw-r--r-- 1 root root 4674 Apr  1 18:00 '^opt^kafka^config^log4j.properties'
     -rw-r--r-- 1 root root   77 Apr  1 18:00 '^opt^kafka^config^log4j.properties.ls'
     ```

3. View the first lines of the file to verify the file contents:

```
# head ^opt^kafka^config^log4j.properties
```

Output:

```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
```

Security bulletins and fixes

Stay informed about known security vulnerabilities and fixes for IBM Cloud Pak® for AIOps by subscribing to the security bulletins and by reviewing the list of fixed security-related vulnerabilities.

- Security bulletins
- Fixed security-related vulnerabilities in version 4.2.1
- Fixed security-related vulnerabilities in previous versions

Security bulletins

Subscribe to IBM Cloud Pak for AIOps notifications by following these steps:

1. Go to the IBM Support site.
2. Scroll to the **Support basics** section. Then, click the **Notification settings** card.
3. Log in to IBM with your IBM ID and password to continue.
4. Enter **IBM Cloud Pak for AIOps** in the **Product lookup** field. Click **Subscribe**.
5. In the **Select document types** page, select **Security bulletin** and **Fixes** > **Security Vulnerability (Sec/Int)**. You can also select any other document types that you need to keep informed about.
6. Click **Submit**.
7. To configure how you receive notifications, click **Delivery preferences** in the banner at the beginning of the page. Edit your settings as needed.

Fixed security-related vulnerabilities in version 4.2.1

Review the following tables, which lists the fixed reported security-related vulnerabilities with IBM Cloud Pak for AIOps, and any included IBM or third-party software.

Table. Fixed Common Vulnerabilities and Exposures in Version 4.2.1

CVE-ID	Issue	Description
CVE-2023-41419	Gevent privilege escalation	Gevent could allow a remote attacker to gain elevated privileges on the system, caused by a flaw in the WSGIServer component. By using a specially crafted script , an attacker could exploit this vulnerability to gain elevated privileges.
CVE-2021-3712	OpenSSL ASN.1 strings information disclosure	OpenSSL could allow a remote attacker to obtain sensitive information, caused by an out-of-bounds read when processing ASN.1 strings. By sending specially crafted data, an attacker could exploit this vulnerability to read contents of memory on the system or perform a denial of service attack.
CVE-2023-3090	Linux Kernel privilege escalation	Linux Kernel could allow a local authenticated attacker to gain elevated privileges on the system, caused by a heap out-of-bounds write flaw due to missing skb->cb initialization in the ipvlan network driver. By sending a specially crafted request, an authenticated attacker could exploit this vulnerability to gain elevated privileges.
CVE-2023-3390	Linux Kernel privilege escalation	Linux Kernel could allow a local authenticated attacker to gain elevated privileges on the system, caused by a use-after-free flaw in the netfilter subsystem in net/netfilter/nf_tables_api.c.. By sending a specially crafted request, an authenticated attacker could exploit this vulnerability to gain elevated privileges.

CVE-ID	Issue	Description
CVE-2023-35001	Linux Kernel privilege escalation	Linux Kernel could allow a local authenticated attacker to gain elevated privileges on the system, caused by a nf_tables nft_byteorder_eval out-of-bounds read/write. By sending a specially crafted request, an attacker could exploit this vulnerability to escalate privileges.
CVE-2023-35788	Linux Kernel privilege escalation	Linux Kernel could allow a local authenticated attacker to gain elevated privileges on the system, caused by an off-by-one flaw in the fl_set_geneve_opt fucntion. By sending a specially crafted request, an authenticated attacker could exploit this vulnerability to gain elevated privileges or cause a denial of service condition.
CVE-2023-3776	Linux Kernel privilege escalation	Linux Kernel could allow a local authenticated attacker to gain elevated privileges on the system, caused by a use-after-free flaw in the net/sched: cls_fw component. By sending a specially crafted request using the reference counter, an authenticated attacker could exploit this vulnerability to gain elevated privileges.
CVE-2023-4004	Linux Kernel privilege escalation	Linux Kernel could allow a local authenticated attacker to gain elevated privileges on the system, caused by a flaw in the nft_pipapo_remove function in the netfilter. By sending a specially crafted request, an authenticated attacker could exploit this vulnerability to gain elevated privileges or cause the system to crash.
CVE-2023-26144	GraphQL GraphQL.js denial of service	GraphQL GraphQL.js is vulnerable to a denial of service, caused by insufficient checks in the OverlappingFieldsCanBeMergedRule.ts. By sending specially crafted queries, a remote attacker could exploit this vulnerability to degrade system performance, and results in a denial of service condition.
CVE-2021-4160	OpenSSL weak security	OpenSSL could provide weaker than expected security, caused by a carry propagation flaw in the MIPS32 and MIPS64 squaring procedure. An attacker could exploit this vulnerability to launch further attacks on the system.
CVE-2023-2002	Linux Kernel command execution	Linux Kernel could allow a remote authenticated attacker to execute arbitrary commands on the system, caused by improper permission check in the Bluetooth subsystem. By sending a specially crafted request, an attacker could exploit this vulnerability to execute arbitrary management commands on the system.
CVE-2023-20593	AMD Ryzen, Gen AMD EPYC Processors information disclosure	AMD Ryzen, Gen AMD EPYC Processors could allow a local authenticated attacker to obtain sensitive information, caused by a use-after-free flaw in the vzeroupper instruction. By conducting a cache timing attack using a specially crafted application, an attacker could exploit this vulnerability to obtain sensitive data used by other processes, such as passwords and encryption keys, at a rate of 30KB/sec from each CPU core.
CVE-2023-32002	Node.js security bypass	Node.js could allow a remote attacker to bypass security restrictions, caused by the use of Module._load(). By sending a specially crafted request, an attacker could exploit this vulnerability to bypass the permission policy mechanism.
CVE-2019-19316	Terraform disclosure	Terraform could allow a remote attacker to obtain sensitive information, caused by transmitting data via cleartext HTTP when using the Azure backend with a shared access signature (SAS). By sniffing the network, a remote attacker could exploit this vulnerability to obtain sensitive information.
CVE-2023-32006	Node.js security bypass	Node.js could allow a remote attacker to bypass security restrictions, caused by the use of module.constructor.createRequire(). By sending a specially crafted request, an attacker could exploit this vulnerability to bypass the permission policy mechanism.
CVE-2023-32559	Node.js security bypass	Node.js could allow a remote attacker to bypass security restrictions, caused by the use of the deprecated API process.binding(). By sending a specially crafted request, an attacker could exploit this vulnerability to bypass the permission policy mechanism.
CVE-2023-33955	MinIO Console information disclosure	MinIO Console could allow a remote authenticated attacker to obtain sensitive information, caused by a flaw when using object names with RIGHT-TO-LEFT OVERRIDE unicode character. By sending a specially crafted request, an attacker could exploit this vulnerability to obtain sensitive information, and use this information to launch further attacks against the affected system.
CVE-2023-27522	Apache HTTP Server HTTP request smuggling	Apache HTTP Server is vulnerable to HTTP request smuggling, caused by improper parsing of the HTTP transfer-encoding request header by the mod_proxy_uwsgi module. By sending a specially-crafted HTTP(S) transfer-encoding request header, an attacker could exploit this vulnerability to poison the web cache, bypass web application firewall protection, and conduct XSS attacks.
CVE-2023-4680	HashiCorp Vault and Vault Enterprise information disclosure	HashiCorp Vault and Vault Enterprise could allow a remote authenticated attacker to obtain sensitive information, caused by a flaw with allowing nonce specified without convergent encryption in the transit secrets engine. By utilize cryptographic attack techniques, an attacker could exploit this vulnerability to obtain sensitive information, and use this information to launch further attacks against the affected system.

Fixed security-related vulnerabilities in previous versions

Review the following documentation, which includes the list of fixed reported security-related vulnerabilities in previous versions of IBM Cloud Pak for AIOps:

- Fixed security-related vulnerabilities in version 4.2.0
- Fixed security-related vulnerabilities in version 4.1.2
- Fixed security-related vulnerabilities in version 4.1.1
- Fixed security-related vulnerabilities in version 4.1.0
- Fixed security-related vulnerabilities in version 3.7.2

- Fixed security-related vulnerabilities in version 3.7.1
- Fixed security-related vulnerabilities in version 3.7.0
- Fixed security-related vulnerabilities in version 3.6.2
- Fixed security-related vulnerabilities in version 3.6.1
- Fixed security-related vulnerabilities in version 3.6.0
- Fixed security-related vulnerabilities in version 3.5.1
- Fixed security-related vulnerabilities in version 3.5.0
- Fixed security-related vulnerabilities in version 3.4.2
- Fixed security-related vulnerabilities in version 3.4.1
- Fixed security-related vulnerabilities in version 3.4.0
- Fixed security-related vulnerabilities in version 3.3.2

Notices

These legal notices pertain to the IBM Cloud Pak® for AIOps product and its documentation.

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any third party product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing Legal and Intellectual Property Law IBM Japan Ltd. 19-21, Nihonbashi-Hakozakicho, Chuo-ku Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information here; these changes will be incorporated in new editions of the publication. IBM may make improvements or changes, or both, in the products or the programs, or both, described in this publication at any time without notice.

Any references in this information to third party websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials on those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information, which has been exchanged, should contact:

IBM Director of Licensing IBM Corporation North Castle Drive, MD-NC119 Armonk, NY 10504-1785 US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning third party products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to third party products. Questions on the capabilities of third party products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE: This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM®, the IBM logo, and ibm.com® are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at *Copyright and trademark information* at www.ibm.com/legal/copytrade.shtml.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

ITIL is a Registered Trade Mark of AXELOS Limited.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and its affiliates, or both.

Red Hat®, JBoss®, OpenShift®, Fedora®, Hibernate®, Ansible®, CloudForms®, RHCA®, RHCE®, RHCSA®, Ceph®, and Gluster® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

VMware, the VMware logo, VMware Cloud Foundation, VMware Cloud Foundation Service, VMware vCenter Server, and VMware vSphere are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Terms and conditions

Personal use

You can reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside of your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the preceding instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Licensing

For full details, see the IBM License Information documents page.

Glossary

This glossary provides terms and definitions for IBM Cloud Pak® for AIOps.

The following cross-references are used in this glossary:

- *See* refers you from a nonpreferred term to the preferred term or from an abbreviation to the spelled-out form.
- *See also* refers you to a related or contrasting term.

While IBM values the use of inclusive language, terms that are outside of IBM's direct influence are sometimes required for the sake of maintaining user understanding. As other industry leaders join IBM in embracing the use of inclusive language, IBM will continue to update the documentation to reflect those changes.

A

accelerator

A set of assets that enable customers to understand core concepts and implement solutions.

AI model

In a machine learning context, a set of functions and algorithms that are trained and tested on a data set to provide predictions or decisions. See also training definition training definition.

alert

A record of an event indicating a fault in the managed environment.

anomaly

A deviation from the expected baseline behaviors.

application

One or more computer programs or software components that provide a function in direct support of a specific business process or processes.

application mapping

Used to expose the on-premises IBM Cloud Pak for AIOps instance to the public cloud. After the application mapping is created, IBM Cloud Pak for AIOps can be accessed from Slack.

auto-remediation

The process of resolving an issue with minimal or no human intervention.

B

boot node

The OpenShift cluster node that is used for running installation, configuration, node scaling, and cluster updates.

C

capability

The core functions of a service. For example, the XYZ service provides "monitoring" capability.

CD

See continuous delivery.

change risk

The risk associated with implementing any type of change, whether it is to code, configurations, or data. See also risk score.

CI

See continuous integration.

connector

The means by which a data source is connected to a product or service. See also integration.

continuous delivery (CD)

A software development practice that employs techniques such as continuous testing, continuous integration, and continuous deployment so that new features and fixes are packaged and deployed rapidly and at low risk to test environments and then to customers.

continuous integration (CI)

A software development practice where members of a team integrate their work frequently so that there are multiple integrations each day. Integrations are verified by an automated build to detect integration errors as quickly as possible.

coverage

A qualitative determination of the AI's ability to produce insights from your incoming data flow as a result of it's training. See also training definition.

CRD

See custom resource definition.

custom resource

An instance of a custom resource definition. See also custom resource definition.

custom resource definition (CRD)

A customizable YAML file that defines a logically related group of objects in a cluster. Custom resource definitions enable a custom resource to be used like any native Kubernetes object in the cluster. See also custom resource.

D

data set

Data that is extracted and organized from data streams. Data sets serve as inputs for pipelines to train, retrain, or tune models.

data stream

A set of processes that convert extracted data into a normalized data set.

E

edge

The relationship, or link, between resources. See also hop, seed.

error

The fault in an event that causes a failure.

event

A change in state of a service or system.

F

failure

The surfacing of an error to the user.

fault

An error in the internal state of a component in a system.

field mapping

The assignment of values from one integration type to a standard set of values. When one maps source fields, one replaces or supplements those values with new or different values.

H

hop

A step along a single edge from one resource to another. See also edge, seed.

I

incident

An unplanned interruption that causes, may cause, or reduces the quality of an IT service. See also incident.

integration

Any independent application that provides additional capabilities to an IBM service or base product. See also connector.

issue

A cause or potential cause of one or more recurring or similar tickets or incidents.

L

log

Data output from an application or service that describes a transaction or state occurring in that application or service.

M

mean time between failures (MTBF)

The average time between failures.

mean time to failure (MTTF)

The average amount of time before a system lapses into a failed state.

mean time to detect (MTTD)

The average amount of time it takes to identify a failure.

mean time to resolve (MTTR)

The total time period from the start of a failure to when the failure resolves and operations resume.

O

observer

A service that extracts resource information and provides topology data for connections and localization and blast radius information for events published to ChatOps connections.

observer job

An automated scheduled task that retrieves topological details for a target system.

operand

The service component used by an operator to perform actions.

operator

A system design that simulates user behavior by linking a controller to custom resources, simulating the act of observing differences in states then acting on them.

P

pipeline

A set of iterative processes that transforms data into insights. For example, retraining pipelines take additional training data sets and improves existing models based on new insights.

proactive channel

A ChatOps channel that is used for change risk notifications. See also reactive channel.

probable cause

A rank-based indicator of causal likelihood that is based on the analysis of alerts and associated topology within the context of an incident.

problem

See issue.

R

reactive channel

A ChatOps channel that is used for outgoing incidents and alerts. See also proactive channel.

resource

A physical or logical component that can be provisioned or reserved for an application or service instance. Examples of resources can include storage, processors, memory, clusters, and VMs.

risk score

A value derived to describe risk across several dimensions associated with change. See also change risk.

RPO (Recovery point objective)

The application loss tolerance, such as the amount of data that can be lost before significant harm to the application occurs.

RTA (Recovery Time Actual)

The actual time that is required for an application to recover completely.

RTO (Recovery Time Objective)

The time that an application can be down without causing significant damage. RTA should be less than RTO.

S

secure tunnel

A feature to package TCP requests and responses, encrypt them with HTTPS, and transport the payloads between IBM Cloud Pak for AIOps and Slack. With Secure Tunnel, you can integrate IBM Cloud Pak for AIOps with Slack without creating firewall rules and policies.

seed

A single resource that serves as the starting point of a topology. Once defined, the topology view is expanded one hop at a time. See also edge, hop.

service

In IBM Cloud Pak solutions, a set of capabilities that provide functionality to a product.

service catalog

A curated repository of services that users can discover and instantiate for use within their organization's applications.

incident

A collection of insights derived from different data sources (logs, events, and alerts) that represent the products' determination of an incident. Incidents build understanding and drive remediation.

T

ticket

A formal record of identified issues or requests that are created against an item and assigned to appropriate users to resolve those issues or complete the requests.

topology

An arrangement of specific, interconnected assets and resources within a network or application.

topology template

A defined topology pattern that is used to dynamically generate topologies based on matching conditions from previously established topologies in the topology database.

training definition

The established model that serves as the baseline for your data to be measured against. For example, your AI model for log anomalies acts as the training definition for all of your log anomaly connections. See also AI model, coverage.

tunnel worker

Server side of a TCP-over-HTTPS tunnel. It is developed based on OSS project, and is installed in the IBM Cloud Pak for AIOps cluster.

tunnel connection

Used to manage IBM Cloud Pak for AIOps instance that you want to expose. If you want to expose on-premises IBM Cloud Pak for AIOps for Slack connection, first you need to create a tunnel connection.

tunnel connector

Client side of a TCP-over-HTTPS tunnel. It is developed based on OSS project, and can be installed in an OpenShift cluster. Ensure that the OpenShift cluster can be accessed from the public network, and the cluster domain certificate is signed.