IBM Cloud Orchestrator
Version 2.4

User's Guide

IBM
# Contents

**Preface**  
Who should read this information  

**Chapter 1. Overview**  
What is new in this release  
Product architecture  
Product features  
Pattern Engines  
Overview of OpenStack  
Multitenancy overview  
Custom extensions  
Deployment modes  

**Chapter 2. Installing**  
Planning your installation  
Installation overview  
IBM Cloud Orchestrator components  
IBM Cloud Orchestrator Enterprise Edition installation  
IBM Platform Resource Scheduler installation  
Deployment topologies  
Hardware prerequisites  
Software prerequisites  
Planning networks  
Using a virtual machine as network server  
Summary of features and capabilities of a network server  
Neutron scenarios  
Preparing the installation media  
Preparing resources to install IBM Cloud Orchestrator  
Setting up virtual machines  
Installing the Deployment Service  
Changing the Deployment Service password  
Understanding deployment templates  
Customizing deployment parameters  
Deployment parameters  
Configuring an external database  
Deploying an IBM Cloud Orchestrator environment  
Using the Deployment Service wizard to deploy IBM Cloud Orchestrator  
Using the command-line interface to deploy IBM Cloud Orchestrator  
Deploying a Region Server  
Installing a KVM Region Server  
Installing a VMware Region Server  
Installing a PowerVC Region Server  
Post PowerVC Region Server installation tasks.  
Installing a z/VM Region Server  
Customizing a z/VM Region Server deployment  
System files modified by the installation procedure  
Deploying the Distributed Active-Active (High Availability) topology  
Installing System Automation Application Manager  
Installing the Central Servers  
Installing the Region Server  
Installing a KVM Region Server with Neutron network  
Installing a KVM Region Server with Nova network  
Installing a VMware Region Server with Neutron network  
Installing a VMware Region Server with Nova network  
Verifying the installation  
Troubleshooting installation  
All-in-one deployment failed  
Preparing to install additional Enterprise Edition components  
Quick start guide for metering and billing  
Performing post-installation tasks  
Assigning zones to domains and projects  
Setting NTP servers  
Configuring Memcached cache size  
Replacing the existing certificates  
Maintaining a deployed environment  
Scaling out a deployed environment  
Back up and restoring Deployment Service  
Move the database to an external system  
Content pack installation  
Configuring Central Nodes to connect to deployed virtual machines  
Open VNC port on ESXi host to enable the VNC console access  
Managing flavors  
Creating new flavors in OpenStack  
Customizing flavors for Power features  
Managing Nova networks  
Adding Nova networks  
Associating a Nova network to a project  
Managing Neutron networks  
Adding Neutron networks  
Scenario 1: Use network server as gateway  
Scenario 2: Use a physical gateway  
Scenario 3: Hybrid SDN and physical routing  
Associating a Neutron network to a project  
Creating multiple networks assigned to different projects  
Neutron quota configuration  
Initializing the IaaS toolkit database  
Performing advanced configuration tasks  

© Copyright IBM Corp. 2013, 2014
# Chapter 3. Administering

Starting or stopping IBM Cloud Orchestrator.

Managing IBM Cloud Orchestrator services manually.

Restarting NoSQL.

Managing high availability.

High-availability solutions.

Enabling the datastore random selection when booting a virtual machine.

Enabling SRDS and Resource Pool.

Connecting to multiple clusters.

Connecting to multiple vCenters.

Port group validation when using nova network.

Enable the datastore random selection when booting a virtual machine.

Configuring vCenter discovery.

Configuring vCenter discovery for multiple vCenters or cluster/resource pool.

Configuring OpenStack to support thin provisioning.

Removing a KVM compute node.

Removing a region.

Configuring LDAP authentication independently of the domain.

Configuring LDAP authentication on a domain-by-domain basis.

Using external Directory Integration tools with LDAP authentication.

LDAP Keystone configuration file reference.

Changing the various passwords.

Starting and stopping the services in the deployment service node.

Configuring high availability.

Configuring the external database server for the high-availability topology.

Enabling autostart of System Automation Application Manager.

Configuring System Automation Application Manager.

Post-installation configurations.

Upgrading from SmartCloud Orchestrator V2.3 or V2.3.0.1.

Upgrading with an external database.

Configuring the upgrade configuration file.

Requirements for upgrading a VMware Region.

Migrating data in a VMware region.

Upgrade troubleshooting.

Upgrading from SmartCloud Cost Management 2.1.0.3.

Post configuration after Public Cloud Gateway is upgraded.

Advanced post configuration after VMware region is upgraded.

Ports used by IBM Cloud Orchestrator.

---

Table of Contents

<table>
<thead>
<tr>
<th>Task</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring high availability</td>
<td>145</td>
</tr>
<tr>
<td>Deploying multiple Cinder nodes as Storage Nodes</td>
<td>117</td>
</tr>
<tr>
<td>Advanced configuration on VMware region</td>
<td>117</td>
</tr>
<tr>
<td>Enabling SRDS and Resource Pool</td>
<td>117</td>
</tr>
<tr>
<td>Connecting to multiple clusters</td>
<td>118</td>
</tr>
<tr>
<td>Connecting to multiple vCenters</td>
<td>119</td>
</tr>
<tr>
<td>Port network validation when using nova</td>
<td>123</td>
</tr>
<tr>
<td>Enable the datastore random selection when booting a virtual machine</td>
<td>123</td>
</tr>
<tr>
<td>Configuring vCenter discovery</td>
<td>123</td>
</tr>
<tr>
<td>Configuring vCenter discovery for multiple vCenters or cluster/resource pool</td>
<td>126</td>
</tr>
<tr>
<td>Configuring OpenStack to support thin provisioning</td>
<td>127</td>
</tr>
<tr>
<td>Removing a KVM compute node</td>
<td>127</td>
</tr>
<tr>
<td>Removing a region</td>
<td>128</td>
</tr>
<tr>
<td>Configuring LDAP authentication independently of the domain</td>
<td>129</td>
</tr>
<tr>
<td>Configuring LDAP authentication on a domain-by-domain basis</td>
<td>133</td>
</tr>
<tr>
<td>Using external Directory Integration tools with LDAP authentication</td>
<td>133</td>
</tr>
<tr>
<td>LDAP Keystone configuration file reference</td>
<td>135</td>
</tr>
<tr>
<td>Changing the various passwords</td>
<td>139</td>
</tr>
<tr>
<td>Starting and stopping the services in the deployment service node</td>
<td>144</td>
</tr>
<tr>
<td>Configuring high availability</td>
<td>145</td>
</tr>
<tr>
<td>Configuring the external database server for the high-availability topology</td>
<td>145</td>
</tr>
<tr>
<td>Enabling autostart of System Automation Application Manager</td>
<td>147</td>
</tr>
<tr>
<td>Configuring System Automation Application Manager</td>
<td>148</td>
</tr>
<tr>
<td>Post-installation configurations</td>
<td>151</td>
</tr>
<tr>
<td>Upgrading from SmartCloud Orchestrator V2.3 or V2.3.0.1</td>
<td>152</td>
</tr>
<tr>
<td>Upgrading with an external database</td>
<td>158</td>
</tr>
<tr>
<td>Configuring the upgrade configuration file</td>
<td>159</td>
</tr>
<tr>
<td>Requirements for upgrading a VMware Region</td>
<td>164</td>
</tr>
<tr>
<td>Migrating data in a VMware region</td>
<td>165</td>
</tr>
<tr>
<td>Upgrade troubleshooting</td>
<td>167</td>
</tr>
<tr>
<td>Upgrading from SmartCloud Cost Management 2.1.0.3</td>
<td>168</td>
</tr>
<tr>
<td>Post configuration after Public Cloud Gateway is upgraded</td>
<td>169</td>
</tr>
<tr>
<td>Advanced post configuration after a VMware region is upgraded</td>
<td>169</td>
</tr>
<tr>
<td>Ports used by IBM Cloud Orchestrator</td>
<td>172</td>
</tr>
<tr>
<td>Configuring OpenStack to support resource overcommitment</td>
<td>116</td>
</tr>
<tr>
<td>Deploying multiple Cinder nodes as Storage Nodes</td>
<td>117</td>
</tr>
<tr>
<td>Advanced configuration on VMware region</td>
<td>117</td>
</tr>
<tr>
<td>Enabling SRDS and Resource Pool</td>
<td>117</td>
</tr>
<tr>
<td>Connecting to multiple clusters</td>
<td>118</td>
</tr>
<tr>
<td>Connecting to multiple vCenters</td>
<td>119</td>
</tr>
<tr>
<td>Port network validation when using nova network</td>
<td>123</td>
</tr>
<tr>
<td>Enable the datastore random selection when booting a virtual machine</td>
<td>123</td>
</tr>
<tr>
<td>Configuring vCenter discovery</td>
<td>123</td>
</tr>
<tr>
<td>Configuring vCenter discovery for multiple vCenters or cluster/resource pool</td>
<td>126</td>
</tr>
<tr>
<td>Configuring OpenStack to support thin provisioning</td>
<td>127</td>
</tr>
<tr>
<td>Removing a KVM compute node</td>
<td>127</td>
</tr>
<tr>
<td>Removing a region</td>
<td>128</td>
</tr>
<tr>
<td>Configuring LDAP authentication independently of the domain</td>
<td>129</td>
</tr>
<tr>
<td>Configuring LDAP authentication on a domain-by-domain basis</td>
<td>133</td>
</tr>
<tr>
<td>Using external Directory Integration tools with LDAP authentication</td>
<td>133</td>
</tr>
<tr>
<td>LDAP Keystone configuration file reference</td>
<td>135</td>
</tr>
<tr>
<td>Changing the various passwords</td>
<td>139</td>
</tr>
<tr>
<td>Starting and stopping the services in the deployment service node</td>
<td>144</td>
</tr>
<tr>
<td>Configuring high availability</td>
<td>145</td>
</tr>
<tr>
<td>Configuring the external database server for the high-availability topology</td>
<td>145</td>
</tr>
<tr>
<td>Enabling autostart of System Automation Application Manager</td>
<td>147</td>
</tr>
<tr>
<td>Configuring System Automation Application Manager</td>
<td>148</td>
</tr>
<tr>
<td>Post-installation configurations</td>
<td>151</td>
</tr>
<tr>
<td>Upgrading from SmartCloud Orchestrator V2.3 or V2.3.0.1</td>
<td>152</td>
</tr>
<tr>
<td>Upgrading with an external database</td>
<td>158</td>
</tr>
<tr>
<td>Configuring the upgrade configuration file</td>
<td>159</td>
</tr>
<tr>
<td>Requirements for upgrading a VMware Region</td>
<td>164</td>
</tr>
<tr>
<td>Migrating data in a VMware region</td>
<td>165</td>
</tr>
<tr>
<td>Upgrade troubleshooting</td>
<td>167</td>
</tr>
<tr>
<td>Upgrading from SmartCloud Cost Management 2.1.0.3</td>
<td>168</td>
</tr>
<tr>
<td>Post configuration after Public Cloud Gateway is upgraded</td>
<td>169</td>
</tr>
<tr>
<td>Advanced post configuration after a VMware region is upgraded</td>
<td>169</td>
</tr>
<tr>
<td>Ports used by IBM Cloud Orchestrator</td>
<td>172</td>
</tr>
</tbody>
</table>
Managing projects. 222
  Creating a project 222
  Enabling a project 223
  Editing a project 223
  Disabling a project 224
  Deleting a project 224
  Assigning a zone to a project 224
  Configuring project quotas 225
  Reassigning VM instances to a project 225
  Customizing and extending the user interface 226
  Modifying user assignments for a project 227
Managing groups 228
  Deleting a group 228
Managing users 228
  Creating a user 229
  Deleting a user 229
Managing volumes 229
Managing networks 229
  Creating a network 230
  Deleting a network 231
  Modifying a network 231
  Adding a subnet to an existing network 231
Using the Public Cloud Gateway 232
  Public Cloud Gateway overview 232
  Capabilities and limitations 232
  OpenStack API support 234
Configuring the Public Cloud Gateway:
  Overview 235
  SSH key management 236
  Multi-tenancy support 236
  Quota support overview 237
Upgrading 238
  Upgrading and cache configuration 238
  Upgrading Amazon EC2 multi-tenancy 238
Performing post-upgrade configuration tasks 239
Common configuration tasks 240
Prerequisites 240
  Creating a supported image 241
  Creating cloud-init images for nova deployment for Linux 241
  Configuring the Public Cloud Gateway for SoftLayer 242
  Tagging images with the configuration strategy 244
  Configuring flavors 245
  Configuring quotas 248
  Configuring caching 248
  Changing keystone administrator password 250
  Changing a region name 250
  Restarting the Public Cloud Gateway 252
Managing Amazon EC2 using the Public Cloud Gateway 252
  Configuring Public Cloud Gateway for Amazon EC2 252
  Configure regions in config.json 253
Managing SoftLayer using the Public Cloud Gateway 256
  Integrating SoftLayer using the Public Cloud Gateway 256
Chapter 4. Managing orchestration workflows 287
  Orchestration workflows 287
  Self-service offerings 288
  Orchestration actions for Virtual System (Classic) Patterns 288
  User actions 289
  Event-triggered actions 289
  Samples and standard extensions for orchestration workflows 290
Working with Business Process Manager 290
  Setting up IBM Process Designer 291
  Adding users to IBM Process Designer 291
Creating a process application in Process Designer 291
  Reusing processes and human services in a process application 293
  Editing process applications and toolkits 293
  Creating a process 293
  User input required at service request time 295
Troubleshooting 267
  Loss of functionality in Public Cloud Gateway cloud groups 267
  Region names displayed incorrectly in the Virtual Image page 267
  Unable to connect to a public cloud due to missing credentials 268
  Unable to deploy instance to non-IBM supplied OpenStack 269
  Inconsistencies in deployed instance names 269
  MAC address field is empty 269
  Quota troubleshooting 270
  Failure to generate admin token 271
Reference 272
  Key pairs 272
  Command-line interface scripts 272
  Password authentication on Amazon EC2 images 272
  Configuring the Public Cloud Gateway regions to use a proxy server 273
Auditing patterns 274
  Audit events 275
  Audit event record attributes and usage tips 276
  Download options for audit event records 279
  Retrieving audit data with the REST API 280
  Deleting audit data to free storage 284
Password management 286
  Changing password 286
  Resetting a password 286
Configuring the Public Cloud Gateway for SoftLayer 257
Managing non-IBM supplied OpenStack using the Public Cloud Gateway 261
  Configure the Public Cloud Gateway to manage non-IBM supplied OpenStack 262
  Configure the Public Cloud Gateway regions for non-IBM supplied OpenStack 262
  Configure non-IBM supplied OpenStack EC2 credentials 263
Performing post-configuration tasks 266
Contents v
Making a new process available as a self-service offering .............. 295
Making a process available as an orchestration action ............... 296
Upgrading a process on a development system or production system .......... 297
Configuring development mode ..... 298
Configuring production mode .... 298
Guidelines for working with Business Process Manager ......... 299
Guidelines for naming and documenting your toolkit or process application.... 299
Guidelines for creating artifacts in a toolkit guidelines to structure your solution .... 301
Guidelines for handling errors...... 301
Chapter 5. Working with self-service 303
Using self-service ............ 303
Viewing the dashboard ......... 303
Submitting a self-service request...... 305
Viewing the status of your requests ..... 305
Managing resources ........... 306
Resource types ................. 306
Working with resources ...... 307
Searching in Assigned Resources. . . . . . . 307
Sorting in Assigned Resources. . . . . . . 307
Applying an action to a resource. . . . . . 307
Managing Virtual Machines ........ 308
Deploying a virtual machine ..... 308
Managing virtual machine instances .... 309
Managing Heat stacks .......... 311
Deploying a Heat stack ....... 311
Managing Heat stack instances ... 314
Managing key pairs ............ 315
Registering a key pair .......... 315
Unregistering a key pair ....... 315
Viewing the status of actions ..... 316
Working with patterns ........... 317
Managing the Inbox .......... 317
Viewing the Inbox. .......... 317
Processing an Inbox assignment .... 317
Managing self-service offerings ..... 318
Self-Service Catalog default contents. 318
Managing offerings ............. 319
Creating an offering .......... 319
Editing an offering .......... 319
Deleting an offering .......... 320
Modifying access control of an offering .. 320
Managing categories ............ 321
Creating a category .......... 321
Editing a category .......... 321
Deleting a category .......... 321
Managing actions ............... 322
Creating an action .......... 322
Editing an action .......... 322
Deleting an action .......... 323
Modifying access control of an action .. 323
Chapter 6. Managing virtual images 325
Creating base images ........ 325
Adding cloud-init to Linux images .... 326
Adding cloud-init to Windows images .... 327
Installing virtio driver (KVM hypervisor only) .... 327
Running sysprep.exe ....... 328
Adding cloud-init to images for Linux on Power 329
Adding cloud-init to images for Linux on System z .... 329
Creating images to deploy via virtual system patterns or virtual application patterns . 330
Software prerequisites for Linux images (KVM or VMware hypervisors) .... 330
Software prerequisites for Microsoft Windows images ... 331
Software prerequisites for images for Linux on System z .... 331
Software prerequisites for AIX images .... 332
Adding images to your OpenStack environment 332
Adding images to IBM Cloud Orchestrator .... 334
Using OVA images from IBM Cloud Orchestrator Catalog on KVM or VMware .... 335
Adding an OVA image to OpenStack .... 336
Mapping the image created in IBM Cloud Orchestrator .... 336
Using OVA images from IBM Cloud Orchestrator Catalog on PowerVC regions .... 338
Making PowerVC images compatible with Workload Deployer .... 339
Using images created for SmartCloud Orchestrator V2.3 .... 340
Creating SmartCloud Orchestrator V2.3-like images (advanced users only) .... 341
Chapter 7. Managing and deploying virtual patterns. ....... 343
Virtual pattern types ........... 343
Managing environment profiles .... 344
Environment profiles overview .... 344
Managing environment profiles in the user interface .... 345
Creating an environment profile .... 345
Cloning an environment profile .... 349
Editing an environment profile .... 350
Environment Profiles window fields .... 351
Managing script packages ....... 354
Script packages overview ....... 355
Using script packages with the user interface .... 356
Adding a script package ....... 357
Cloning a script package ....... 360
Making script packages read-only .... 360
Associating a script package with a pattern .... 361
Deleting a script package ....... 363
Configuring a script package using a JSON object .... 364
Script package environment variables .... 365
Properties variable syntax ....... 367
Script package examples ....... 368
Example script package to install an application .... 380
Example script to configure the trace levels .... 382
Example script package to create a server .... 383
Developing plug-ins . 551
Plug-in Development Kit . 556
Plug-in development guide . 560
Samples . 604
Deploying virtual applications . 612
Deploying virtual application patterns . 612
Deploying virtual application templates . 616
Securing virtual applications . 618
Securing virtual application instances with SSL . 619
Configuring SSH key-based access during application deployment . 621
Configuring SSH key-based access in the user interface . 622
LTPA keys . 623
Working with virtual application instances . 624
Monitoring virtual application instances . 626
Viewing virtual application instance logs . 626
Working with shared services . 628
Deploying shared services . 628
Monitoring shared services . 629
Deploying a monitoring shared service . 630
Enabling TOSCA support for IBM Cloud Orchestrator . 631
Configuring the RPM repository for the deployment of TOSCA patterns . 632
Configuring an RPM repository of type ISO Image on NFS . 632
Configuring an RPM repository of type FTP . 633
Configuring an RPM repository of type HTTP . 633
Disabling the configuration of an RPM repository . 634
Configuring Chef support for TOSCA based virtual application patterns . 634

Chapter 8. Integrating . 637
Integrating with IBM Tivoli Monitoring . 637
Preparing a base operating system . 637
Database setup . 638
Installing IBM Tivoli Monitoring . 638
Packages used for installation . 639
Creating a warehouse database . 640
Monitoring Agent for Linux . 641
Monitoring Agent for Kernel-based virtual machines . 641
OpenStack hypervisors . 642
Integrating with IBM Endpoint Manager . 642
Installing Endpoint Manager agent to existing computers . 643
Installing Endpoint Manager agents during virtual machine deployment . 643

Chapter 9. Reporting . 653
Tivoli Common Reporting . 653

Chapter 10. Reference . 655
REST API reference . 655
REST API frameworks . 656
Orchestration actions REST API . 656
List all orchestration action entries . 656
Retrieve an orchestration action entry . 658
Add or update an entry in the orchestration actions . 658
Delete an orchestration action entry . 659
Get entries for a specific virtual system . 660
Business Process Manager Invoker REST API . 661
Retrieve available BPM Business Processes . 661
List all BPM Business Processes . 661
Get entries for a specific BPM Business Process . 662
Retrieve available human services . 662
List all human services . 662
Get entries for a specific human service . 663
Retrieve the Inbox . 664
List all Inbox items . 664
Get entries for a specific Inbox item . 666
Service instance REST API . 667
Get entries for a specific service instance . 668
Metadata parameters REST API . 671
Get parameters of specific metadata . 671
Post metadata parameters . 672
Delete parameters of specific metadata . 672
Virtual machines parameters REST API . 673
Get parameters of a specific virtual machine . 673
Post virtual machine parameters . 673
Delete parameters of specific virtual machines . 674
Deployment parameters REST API . 674
Get deployment parameters for a specific instance . 674
Post deployment parameters for a specific instance . 675
Core Services REST API . 676
Core Services REST API overview . 676
Offering REST API V2 . 679
Categories . 679
Offering attributes . 682
Offering instances . 684
Resource Instances REST API . 689
Resource Instance Providers . 696
Task Engine REST API V2 . 704
Configuration Providers REST API . 707
Managing entities using Core REST APIs . 707
Managing entities using actions . 708
Core REST API for compatibility with earlier versions . 713
Self-service offering REST API . 713
Self-service catalog REST API . 727
Task engine REST API . 733
Workload Deployer REST API . 736
Application patterns REST API . 736
Environment profiles REST API . 745
Log viewer manager REST API . 748
Logging REST API . 750
Monitoring REST API . 752
Pattern types REST API . 756
Patterns REST API . 759
Plug-ins REST API . 763
Shared services REST API . 765
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version REST API</td>
<td>767</td>
</tr>
<tr>
<td>Virtual appliance instances REST API</td>
<td>768</td>
</tr>
<tr>
<td>Virtual applications REST API</td>
<td>772</td>
</tr>
<tr>
<td>Virtual images REST API</td>
<td>777</td>
</tr>
<tr>
<td>Virtual machines REST API</td>
<td>782</td>
</tr>
<tr>
<td>Virtual system instances REST API</td>
<td>787</td>
</tr>
<tr>
<td>Virtual system patterns REST API</td>
<td>798</td>
</tr>
</tbody>
</table>

### Chapter 11. Troubleshooting | 811

- Finding the log files | 811
- Problem determination with pdcollect tool | 812
- Running the pdcollect tool | 813
- Running the pdcollect tool with non-root user | 814
- Setting logging levels | 815
- Workload Deployer log files | 817
- Known errors and limitations | 818
  - Product limitations | 818
  - Unable to reduce the disk size of VMware and KVM virtual machines | 819
  - Hypervisor errors | 820
  - Image errors | 821
  - Instance errors | 823
  - Deployment errors | 824
  - Setting the host name when deploying a Windows system | 824
  - Unable to deploy a virtual system pattern with a non-admin user | 825
  - Error displayed when deploying a virtual system pattern | 826
  - Unable to deploy WebSphere Application Server OVA image | 826
  - Deployment of the virtual system pattern fails due to the name of the virtual machine | 826
  - Script execution does not report failing condition | 827
  - No valid host was found | 828
- nova command errors and limitations | 828
- Security limitations | 828
- Unable to change the flavor of VMware virtual machines | 829
- Error message displayed when trying to start a virtual system | 829
- Unable to list keystone endpoints on the Region Server | 830
- Unable to log in by using an LDAP user | 831
- Unable to add disk to SLES instance | 831
- Errors of 32-bit library files libstdc++.so.6 and /lib/libpam.so* were not found in db2prereqcheck.log | 831
- Unable to reach one of the addresses if multiple NICs of a Linux virtual machine are deployed | 832
- Unable to access the Administration user interface | 833
- User name and password for vCenter are incorrect | 833
- Hypervisors remain in maintenance mode | 834
- Unable to list all the existing resources | 834
- Cannot configure one nova compute service to connect too many vCenter clusters ( >4 ) | 835
- Failed to start a virtual system instance on a KVM Region | 835
- Troubleshooting for a VMware Region | 836
- Troubleshooting Workload Deployer | 836
- Troubleshooting a high-availability installation | 837
- Troubleshooting Business Process Manager | 840

### Notices | 843

### Trademarks | 845

### Glossary | 847

<table>
<thead>
<tr>
<th>Letter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>847</td>
</tr>
<tr>
<td>B</td>
<td>847</td>
</tr>
<tr>
<td>C</td>
<td>848</td>
</tr>
<tr>
<td>D</td>
<td>848</td>
</tr>
<tr>
<td>E</td>
<td>848</td>
</tr>
<tr>
<td>F</td>
<td>848</td>
</tr>
<tr>
<td>G</td>
<td>848</td>
</tr>
<tr>
<td>H</td>
<td>848</td>
</tr>
<tr>
<td>I</td>
<td>849</td>
</tr>
<tr>
<td>J</td>
<td>849</td>
</tr>
<tr>
<td>K</td>
<td>849</td>
</tr>
<tr>
<td>L</td>
<td>849</td>
</tr>
<tr>
<td>M</td>
<td>849</td>
</tr>
<tr>
<td>N</td>
<td>849</td>
</tr>
<tr>
<td>O</td>
<td>849</td>
</tr>
<tr>
<td>P</td>
<td>849</td>
</tr>
<tr>
<td>Q</td>
<td>849</td>
</tr>
<tr>
<td>R</td>
<td>849</td>
</tr>
<tr>
<td>S</td>
<td>849</td>
</tr>
<tr>
<td>T</td>
<td>849</td>
</tr>
<tr>
<td>U</td>
<td>849</td>
</tr>
<tr>
<td>V</td>
<td>849</td>
</tr>
</tbody>
</table>

### Privacy policy considerations | 851

### Accessibility features for IBM Cloud Orchestrator | 853
Preface

This publication documents how to use IBM® Cloud Orchestrator.

Who should read this information

This information is intended for cloud administrators who install and configure IBM Cloud Orchestrator, and for users who work with this product.
Chapter 1. Overview

With IBM Cloud Orchestrator, you can manage your cloud infrastructure.

IBM Cloud Orchestrator helps you with end-to-end service deployment across infrastructure and platform layers. It also provides integrated IT workflow capabilities for process automation and IT governance, resource monitoring, and cost management. The product offers you an extensible approach to integration with existing environments such as network management tools. It facilitates integration with customer-specific service management processes, such as those defined in the IT infrastructure library (ITIL).

Using IBM Cloud Orchestrator, you have a consistent, flexible, and automated way of integrating the cloud with customer data center policies, processes, and infrastructures across various IT domains, such as backup, monitoring, and security. Use the intuitive, graphical tool in IBM Cloud Orchestrator to define and implement business rules and IT policies. You can connect the aspects of different domains into a consistent orchestration of automated and manual tasks to achieve your business goals.

IBM Cloud Orchestrator is based on a common cloud platform that is shared across IBM’s Cloud offerings. This common cloud stack provides a common realization of the core technologies for comprehensive and efficient management of cloud systems.

You choose between two editions: IBM Cloud Orchestrator and IBM Cloud Orchestrator Enterprise Edition which also includes Monitoring and Cost Management.

What is new in this release

The following enhancements were introduced in the current release.

**Simplified architecture**

Image Construction and Composition Tool, Virtual Image Library, IaaS gateway, and SmartCloud Entry are no longer included in IBM Cloud Orchestrator. This implies a simplified procedure to adapt or create images to be deployed via IBM Cloud Orchestrator and a simplified image management procedure.

**Monitoring dashboard**

It provides basic monitoring information related to your infrastructure.

**Simplified single virtual machine deployment**

Deploy single instances without creating a virtual system pattern.

**Support for OpenStack Heat templates and stack**

Import OpenStack Heat templates and deploy and manage OpenStack Heat stacks by using the Self-service user interface.

**Enhanced multitenancy**

Delegate roles and access to resources among cloud administrator, domain administrator, service designer, and end user.
Enhanced virtual machines management
Start, stop, resize, images created outside of the IBM Cloud Orchestrator environment and map them to OpenStack projects from the Self-service user interface.

Integration with SoftLayer
Extend your cloud to deploy instances and orchestrate resources on SoftLayer.

New Administration user interface
Manage your virtual infrastructure and user access via the new user interface.

Enhanced metering capabilities
Exploit OpenStack Ceilometer to collect metering data.

Capability to add custom actions to instances
Add orchestration actions directly to instances.

Enhanced high availability of the IBM Cloud Orchestrator management stack
Enhanced high availability is provided by introducing redundancy and improved recovery for core software components of the IBM Cloud Orchestrator management stack.

Built-in orchestration content
IBM Cloud Orchestrator includes content packs that cover the most typical automation scenarios related to infrastructure-as-a-service and platform-as-a-service scenarios.

Product architecture
IBM Cloud Orchestrator is a comprehensive product that integrates the capabilities of several other IBM solutions.

The main components of IBM Cloud Orchestrator are the process engine and the corresponding modeling user interface, which is used to create processes. For this purpose, IBM Cloud Orchestrator uses the capabilities of IBM Business Process Manager. It also integrates other domain-specific components that are responsible for such functions as monitoring, metering, and accounting. IBM Cloud Orchestrator bundles all these products and components and provides processes that are required to implement the domain-specific functionalities.
The following is a description of the role each major component plays in IBM Cloud Orchestrator:

**Infrastructure-as-a-Service**

The Infrastructure-as-a-Service (IaaS) component is responsible for managing access to compute, storage and networking resources in the virtual environment. All requests to provision services across these services is performed by this component. The IaaS component is delivered by using OpenStack, a leading open source, community-driven project for highly scalable, highly resilient cloud infrastructure management. IBM is one of the Platinum Members of the OpenStack Foundation.

**Software stacks**

While not a specific component itself, Software Stacks represent the concept that when one or more virtual systems are deployed, it is also possible to specify multiple software packages to be deployed upon first boot of those systems. It can be done by invoking simple installation scripts, but also other strong tools can be used such as Chef recipes and cookbooks for automated installation and configuration.

**Patterns**

Patterns allow for deploying more complex middleware configurations and multinode applications. The Patterns component provides a graphical editor that allows the user to describe multiple virtual systems, each with a base image and set of software to be installed, and then specify the relationships and configuration scripts necessary to connect those systems together. With this level of automation, an entire multisystem deployment can be done with just a few simple clicks.

**Workflow orchestration**

The Workflow Orchestration component provides a graphical editor that allows the user to easily customize and extend the procedures that are followed when a user request is initiated. In addition, it also provides the facilities to customize the self-service catalog so that users have access to a variety of service request types that they can access. This component is delivered by embedding IBM’s award-winning Business Process Manager technology along with a number of pre-built automation toolkits that make it possible to integrate workflow automation with the cloud platform and
its other components. The graphical designer is highly flexible, providing many integration techniques ranging from invocation of simple scripts and calling web services to invoking more sophisticated programs such as those written in Java.

**IBM Cloud Orchestrator Catalog**

The IBM Cloud Orchestrator Catalog is a publicly accessible website where various forms of automation can be downloaded and used within IBM Cloud Orchestrator. It includes references to supported automation communities such as Chef, ready to use Patterns and Images, and a variety of pre-built Workflow Orchestration routines, packages and toolkits. It is designed to "ship when ready", meaning that new automation can become available at any time, regardless of IBM Cloud Orchestrator release schedules.

**Service management**

This box represents optional additional management functions that are included in IBM Cloud Orchestrator Enterprise Edition. It also highlights the ability to integrate through Workflow Orchestration other management tools and disciplines that may be important within your environment.

**Development tools**

This box represents the ability to integrate developer tools from IBM Rational Team Concert and a set of plug-ins within Cloud Continuous Delivery such as that a user can automate a "continuous delivery pipeline" from check-in of code, through build, deployment, test, and promotion. Those tools are not provided within IBM Cloud Orchestrator, but more information about them can be found on ibm.com.

---

**Product features**

Read about the main features that are available with IBM Cloud Orchestrator version 2.4.

**Pattern-based cloud delivery**

IBM Cloud Orchestrator provides capabilities for multi-domain orchestration (virtual system, virtual application, and OpenStack Heat infrastructural stack) and a graphical orchestrator for simple composition of cloud automation.

IBM Cloud Orchestrator is a private cloud offering based on the open sourced OpenStack software that significantly speeds up and simplifies managing an enterprise-grade cloud. You have a core set of open source-based technologies to build enterprise-class cloud services that can be ported across hybrid cloud environments.

IBM Cloud Orchestrator gives you greater flexibility by removing the need to develop specific interfaces for different cloud services. You can quickly combine and deploy various cloud services onto the cloud infrastructure by lining up the compute, storage and network resources with an easy-to-use graphical interface.

**Designing business processes**

IBM Cloud Orchestrator is integrated with IBM Business Process Manager, a workflow engine with graphical tooling. It provides a possibility to create and edit complex workflows through simple drag and drop. You can extend the capabilities
of IBM Cloud Orchestrator and design your own workflows. For more information see "Custom extensions" on page 10.

Self-service user interface

Intuitive self-service user interface containing a customizable catalog of offerings is available for users. Offerings can be grouped into categories which are created by administrators to fit the needs of your environment. For more information about self-service, see "Managing self-service offerings" on page 318.

Administration user interface

A graphical user interface that allows you to easily manage and monitor your infrastructure. You can define networks, flavors, inspect the actual resource consumption, manage users, roles, and projects. For more information about using the Administration user interface, see "Administering as cloud administrator" on page 218.

TOSCA support

IBM Cloud Orchestrator supports importing, deploying, and exporting service templates according to the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA). This enables the consumption of third-party content provided in a standardized format.

Cost management

The IBM SmartCloud Cost Management component of the Enterprise Edition provides functionality for collecting, analyzing, reporting, and billing that is based on usage and costs of shared computing resources. With this tool, you can understand your costs and track, allocate, and invoice based on allocated or actual resource use by department, user, and many more criteria. For more information about cost management, see "Cost management".

Within IBM Cloud Orchestrator metering is primarily driven from the OpenStack layer to capture all virtual machine provisioning requests. For more information, see the OpenStack Collector topic.

Monitoring

In the Enterprise Edition of IBM Cloud Orchestrator you can monitor workloads and instances using IBM Tivoli Monitoring. With this component, you can measure the cost of cloud services with metering and charge-back capabilities. For more information about monitoring, see "Integrating with IBM Tivoli Monitoring" on page 637.
Pattern Engines

Learn about the different IBM Cloud Orchestrator pattern engines and determine which one is more appropriate for your environment.

OpenStack Heat

OpenStack Heat templates are suitable for scenarios that focus on the infrastructure. You can create resources such as instances, networks, volumes, security groups, users, and floating IP addresses, and define the relationships between these resources (for example, a volume must be attached to a specific instance, some instances are to be connected using this network, and so on). It allows the addition of auto-scaling services integrated with OpenStack Ceilometer. Even though OpenStack Heat can be integrated with Puppet or Chef it is not the recommended engine to create patterns in which software installation and configuration are crucial. Images to be deployed via Heat templates require that cloud-init be installed. For more information about OpenStack Heat, see “Managing Heat stacks” on page 311.

Virtual System Patterns (classic)

Virtual System Patterns (classic) are used for backward compatibility with SmartCloud Orchestrator 2.3. You can deploy multiple instances by adding network interfaces, volumes, users, and groups. You can add and configure software at deployment time. Images to be deployed via virtual system patterns (classic) require the activation engine to be installed. Images that are purchased from IBM Cloud Orchestrator Catalog can also be used. For more information, see Chapter 7, “Managing and deploying virtual patterns,” on page 343, and Chapter 6, “Managing virtual images,” on page 325.

Virtual System Patterns and Virtual Application Patterns

These patterns provide the same functions as the Virtual System Patterns (classic), enhanced with the possibility of defining scaling policies based on resource consumption in the instances, routing policies for HTTP/HTTPS ports, and security policies to disable root ssh access. You can serialize and order software installation and configuration steps across multiple instances. These steps can include operating system reboots. Images to be deployed via Virtual System Patterns and Virtual Application Patterns require that cloud-init and other software prerequisites be installed. For more information, see Chapter 7, “Managing and deploying virtual patterns,” on page 343, and Chapter 6, “Managing virtual images,” on page 325.

The following table explains the differences between the different deployment and pattern engines.

<table>
<thead>
<tr>
<th></th>
<th>Single Server - Nova</th>
<th>Stacks - Heat</th>
<th>Patterns - IBM Workload Deployer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>Small</td>
<td>Medium</td>
<td>Large</td>
</tr>
<tr>
<td>Purpose</td>
<td>Provision a single virtual machine</td>
<td>Provisions multiple virtual machines with network and storage</td>
<td>Provisions multiple virtual machines with network, storage and additional software</td>
</tr>
</tbody>
</table>
### Table 1. (continued)

<table>
<thead>
<tr>
<th>Capabilities</th>
<th>Single Server - Nova</th>
<th>Stacks - Heat</th>
<th>Patterns - IBM Workload Deployer</th>
</tr>
</thead>
</table>
|              | • user id and password or ssh key  
|              | • image and flavor  
|              | • region and availability zone  
|              | • network  
|              | • input yaml file  
|              | • user id and password or ssh key  
|              | • lookup of input parameters  
|              | • add network and storage  
|              | • graphical editor  
|              | • define dependencies and order  
|              | • add software bundles  
|              | • add script executions  
| Actions      | • details view  
|              | • start and stop  
|              | • delete  
|              | • execute script  
|              | • custom actions  
|              | • details view  
|              | • delete  
|              | • single VM actions for parts  
|              | • custom actions  
|              | • graphical details view  
|              | • start and stop  
|              | • delete  
|              | • run user actions  
|              | • custom actions  

| Supported Hypervisions | All | All, except Public Cloud Gateway | All |

---

### Overview of OpenStack

IBM Cloud Orchestrator is based on OpenStack (the Icehouse release).

OpenStack is a collection of open source technologies that provide scalable computing software for both public and private clouds. For detailed information about OpenStack, see the [OpenStack documentation](#).

IBM Cloud Orchestrator uses the following components and services of OpenStack:

- **Image** (codenamed *Glance*) that provides a catalog and repository for virtual disk images. The virtual disk images are mostly used in the OpenStack Compute service component.
- **Compute** (codenamed *Nova*) that provides virtual servers on demand.
- **Identity** (codenamed *Keystone*) that provides authentication and authorizations for all OpenStack services.
- **Block Storage** (codenamed *Cinder*) that provides persistent block storage to guest virtual machines.
- **Network** (codenamed *Neutron*) that provides network management.
- **Dashboard** (codenamed *Horizon*) that provides a web-based user interface.

To configure and administer IBM Cloud Orchestrator, use the OpenStack command-line interface or the Admin Console. For example, you might need to use the *keystone* command-line interface to manage authentication and authorizations, and the *glance* command-line interface to manage virtual images. For more information about the OpenStack command-line interface, see the [OpenStack CLI Guide](#).
Multitenancy overview

This section describes the roles and delegation in IBM Cloud Orchestrator.

Delegation

Delegation means that a more powerful role can delegate certain tasks to a less powerful role. It is distinguished between two different types of personas:

Service provider: responsible to host IBM Cloud Orchestrator and provide the cloud infrastructure and services.

Service consumer: consumes services from the service provider and acts only in the context of the tenant.

User interfaces

IBM Cloud Orchestrator provides different user interfaces for these roles. The user interfaces are optimized for the user experience of that role. The following user interfaces exist:

IBM Cloud Orchestrator Administration
Only used by aCloud Administrator. The User Interface is based on OpenStack Horizon and allows the configuration of the cloud infrastructure and identities. The view shows the resources in the context of a selected region.

IBM Process Designer and IBM Business Process Manager
Is only used by cloud administrators and content developers. The main User Interface to develop new toolkits and catalog content like processes and human services. It can be used to load new content from the IBM Cloud Orchestrator Catalog.

Self-service user interface
Is mainly used by tenant users, like domain administrator, catalog editors and end users. It provides a self-service portal with dashboard, self-service catalog and managing instances owned by the user. It further support configuration of the domain and catalog content, and a bandwidth of panels to manage patterns including a graphical editor.

Roles

User interfaces are used by the different personas. The following list explains the roles, starting from the most powerful, Cloud Administrator, to the most restrictive, End User.

Cloud Administrator: Service provider
The Cloud Administrator is the most powerful role who can manage and administer the whole cloud infrastructure resources, identities, self-service offerings and patterns across all tenants. A special persona of the Cloud Administrator is the content developer who implements content packs, processes and coaches that implement the offerings. The Cloud Administrator can delegate certain tasks, like user, project, catalog, and pattern configuration to the Domain Administrator.

Domain Administrator: Service consumer
The Domain Administrator is the most powerful role within a tenant but
less powerful than the Cloud Administrator. The Domain Administrator is responsible to setup users, projects, catalog content and patterns in the domain. However the Domain Administrator can only rely on resources that are assigned to the domain by the Cloud Administrator. The Domain Administrator can delegate certain tasks like pattern and catalog configuration to the catalog editor.

**Catalog Editor : Service consumer**

The catalog editor is responsible for creating patterns in the cloud and configuring the offerings and actions for users and projects in the domain. The catalog editor relies on the content packs that are exposed to the domain by the Cloud Administrator. For example, the catalog editor can create patterns, implement script packages and define software packages that are deployed with the pattern. It then exposes the patterns via offerings in the service catalog, which are then used by the End User.

**End User : Service consumer**

The End User can request service like virtual machines, pattern instances and stacks, from the service catalog. The End User can also work with the instances, like start, stop virtual machine or install software on virtual machine. Furthermore, the End User can view the dashboard and can work on inbox assignments. The End User always works on-behalf-of a project.

**Custom: Service consumer**

A role with same rights as the End User. The service provider can decide to implement an approval process for customers and could introduce the role **Approver**. The content packs could then use the **Approver** role to assign any approval request to users with that role. However, it would still be in the responsibility of the Domain Administrator to decide which user is the approver of a project.

For more on the role of Cloud Administrator click on the following links:

- [Installing](#)
- Chapter 3, “Administering,” on page 173
- Chapter 9, “Reporting,” on page 653
- Metering and billing
- Chapter 11, ”Troubleshooting,” on page 811

For more on the responsibility of the Domain Administrator go to [Administering as Domain Administrator](#)

For more on the role of service designers click on the following links:

- Managing orchestration workflows
- Working with self-service
- Managing virtual images
- Managing and deploying virtual patterns
- Chapter 7, “Managing and deploying virtual patterns,” on page 343
- Chapter 10, “Reference,” on page 655

For more on the role of an End User go to [Using self-service](#)
Custom extensions

You create custom extensions to IBM Cloud Orchestrator in the Business Process Manager Process Designer tool and base them on Business Process Manager business processes. To implement user interface extensions, you can use Business Process Manager human services.

IBM Cloud Orchestrator delivers a set of Business Process Manager toolkits that cover the most common automation scenarios in the infrastructure-as-a-service and platform-as-a-service environments. Each toolkit provides a set of reusable artifacts:

Business processes
A business process is any course of action or procedure that an organization follows to achieve a larger business goal. When you break it down, a business process is actually a series of individual tasks or activities that are performed in a specific order. Business processes provide the primary means through which enterprise services are integrated.

Services
Services provide functions for a business process, which itself is a sequence of services. Creating services separately from a business process means a service can be developed independently of a business process and that many types of business processes can reuse that service.

Human services
Human service includes an activity in your business process definition that creates an interactive task that process participants can perform in a web-based user interface.

Coaches
Coaches are the user interfaces for human services.

Business object definitions
Business objects carry the functional properties, data transformation information, and file content that the adapter needs to process requests and generate responses.

With the help of these artifacts, you can efficiently build custom extensions for IBM Cloud Orchestrator. The provided toolkits also contain numerous samples that show how to define custom extensions.

You can download more Business Process Manager toolkits from the IBM Cloud Orchestrator Catalog. These toolkits provide more content for different areas, such as networking or storage, and you can also use them to build IBM Cloud Orchestrator extensions.

Restriction: If you define more than one snapshot for Business Process Manager process application or toolkit, you will be able to use only the artifacts of the top level to define a new extension in IBM Cloud Orchestrator.
Deployment modes

IBM Cloud Orchestrator V2.4 supports several deployment modes on a variety of hypervisor types.

IBM Cloud Orchestrator supports deployment in Demo, Distributed, and Distributed Active-Active (High Availability) modes. Optionally, you can also deploy OpenStack Neutron and an external IBM DB2® database.

IBM Cloud Orchestrator V2.4 introduces a new deployment topology to make the management stack highly available. The Distributed Active-Active (High Availability) topology provides redundancy and improved recovery for core software components of the IBM Cloud Orchestrator management stack. The key benefit of the new Distributed Active-Active (High Availability) topology is a reduction of unplanned and planned downtimes. The new topology ensures that, in certain failure situations, the processing of the IBM Cloud Orchestrator management stack is not interrupted, and incoming deployment requests can be processed even though some components of the cloud management stack failed. Examples include the introduction of application clustering for IBM Business Process Manager, OpenStack Keystone, and many other OpenStack components. In addition, an improved recovery approach is introduced by using classic high-availability clustering. This approach is especially useful for software components that cannot run in an Active-Active setup, but allow classic failover scenarios. Another key benefit of the new Distributed Active-Active (High Availability) deployment topology is improved performance. Core IBM Cloud Orchestrator components run in an Active-Active setup, which improves throughput. The Distributed Active-Active (High Availability) topology is installed through the IBM Cloud Orchestrator Deployment Service. The Deployment Service automates most parts of the high-availability installation, and configuration is fully automated, which reduces and simplifies the overall installation process for the IBM Cloud Orchestrator administrator.

For more information about these deployment modes, see “Deployment topologies” on page 16.

IBM Cloud Orchestrator supports the following hypervisor types: KVM, VMware, PowerVC, and z/VM.

Amazon EC2 and SoftLayer are supported via the Public Cloud Gateway. For more information, see “Using the Public Cloud Gateway” on page 232.

IBM Cloud Orchestrator Enterprise Edition provides additional capabilities from IBM Tivoli® Monitoring and IBM SmartCloud Cost Management. For more information about these products, see “Integrating with IBM Tivoli Monitoring” on page 637 and “Metering and billing” on page 637.
Chapter 2. Installing

Follow this procedure to install IBM Cloud Orchestrator.

Planning your installation

Before you start the installation, review the requirements and plan the whole process.

Installation overview

Get familiar with the basic concepts of IBM Cloud Orchestrator so that you can plan your installation.

The main components of a IBM Cloud Orchestrator installation topology are:

Deployment Server
Hosts the Deployment Service that is the deployment management component to deploy a IBM Cloud Orchestrator environment with a predefined topology.

Central Servers
Host the core IBM Cloud Orchestrator management components.

Region Servers
Are the components used to communicate with a specific hypervisor management infrastructure (KVM, VMware, PowerVC, or z/VM). The KVM region server requires one or more KVM compute nodes to provide the compute resources. The VMware Region server needs to connect to existing VMware Virtual Center to provide virtual machines. The PowerVC region server requires to connect to existing PowerVC to provide virtual machines. The z/VM region server needs to connect to xCat management Node on z/VM to provide virtual machines.

KVM Compute Nodes
Are the components used to manage the virtual machines through the interface provided by KVM.

After the deployment, you can also use the Deployment Service to manage your environment in terms of update, scale out, or delete. Additional Region Servers can be added to enable a multiple-region environment.

The first step in the installation procedure is to install the Deployment Service. Then, you install the Central Servers and, as last step, you setup the Region Servers.

Depending on your needs and the available hardware resources, using the predefined topology templates you can set up one of the following environments:
- A demo environment
- An environment with management components spread across multiple nodes
- An environment with management components spread across multiple nodes and high availability
**IBM Cloud Orchestrator components**

Before understanding the installation topologies, it is important to familiarize yourself with the components included in IBM Cloud Orchestrator.

**Administration user interface**
- Based on the OpenStack Horizon technology. You can monitor and manage the infrastructure.

**Business Process Manager**
- The heart of the orchestration engine, it includes tooling and run time for process design, execution, monitoring, and optimization of business processes.

**Database**
- A DB2 database is used to store all the IBM Cloud Orchestrator persistent data. OpenStack services, Workload Deplyeer, and Business Process Manager use this database. A DB2 instance is also used by the deployment server to store installation and configuration data.

**haproxy**
- Provides access to OpenStack services in high availability configurations.

**IBM HTTP Server**
- Used in high availability configuration as load balancer for the Business Process Manager.

**OpenStack ceilometer**
- Collects metering data related to CPU and networking.

**OpenStack cinder**
- Storage block service that provides persistent block storage to guest virtual machines.

**OpenStack glance**
- Image service and it provides a catalog and repository for virtual disk images.

**OpenStack heat**
- Infrastructure orchestration service. It allows you to deploy instances, manage neutron subnets, ports, routers, and gateways.

**OpenStack keystone**
- Identity service that provides authorization and authentication for the other IBM Cloud Orchestrator components.

**OpenStack neutron (optional)**
- Provides network management services for advanced networking scenarios. Basic networking scenarios can be covered using nova network services.

**OpenStack nova**
- Compute service; it provides virtual servers on demand.

**Public Cloud Gateway**
- Responsible for the communication with Amazon EC2 and SoftLayer.

**qpid**
- Messaging system leveraged by the OpenStack services.

**Self-service user interface**
- User interface for the end user. It provides access to the offerings.

**System Automation Application Manager**
- Automates the availability of resources by starting and stopping resources automatically and in the correct sequence. System Automation Application
Manager uses agentless adapters to monitor and control remote applications. It provides a centralized server that monitors and automatically stops, starts, or restarts the various applications on the virtual machines in the IBM Cloud Orchestrator environment.

**System Automation for Multiplatforms**
Clustering solution that provides high-availability and automation for critical IBM Cloud Orchestrator applications as, for example, the haproxy.

**Workload Deployer**
Advanced pattern engine with which you can design and deploy software solutions spread across multiple virtual instances with automatic scalability policies.

**IBM Cloud Orchestrator Enterprise Edition installation**
For more control over your cloud environment, IBM Cloud Orchestrator Enterprise Edition bundles three additional products: Jazz™ for Service Management, IBM Tivoli Monitoring, and IBM SmartCloud Cost Management.

These products provide the following additional functionality:
- Jazz for Service Management: Reporting
- IBM Tivoli Monitoring: Monitoring
- IBM SmartCloud Cost Management: Metering and billing

**Installation flow**
The first part of the IBM Cloud Orchestrator Enterprise Edition installation is exactly the same as for the base version. Then you must install the additional products on separate machines:

1. Refer to the installation procedure in the Chapter 2, “Installing,” on page 13 section to install IBM Cloud Orchestrator and its services.
2. Install Jazz for Service Management V1.1.0.1. For instructions, see the Jazz for Service Management V1.1.0.1 Quick Start Guide.
3. Install IBM Tivoli Monitoring V6.3.0.2. For instructions, see “Installing IBM Tivoli Monitoring” on page 638.
4. [Optional] Install IBM Tivoli Monitoring for Virtual Environments V7.2.0.2. For instructions, see IBM Tivoli Monitoring for Virtual Environments Quick Start Guide.
5. Install IBM SmartCloud Cost Management V2.1.0.4. For instructions, see “Quick start guide for metering and billing” on page 93.

**IBM Platform Resource Scheduler installation**
Platform Resource Scheduler integrates IBM OpenStack with Enterprise Grid Orchestrator (EGO).

OpenStack is used to provision VM instances. EGO is used to schedule resources for OpenStack to make decisions on where to deploy the VM instance with specified resource selection criteria, such as migrating, resizing, and powering on.

Platform Resource Scheduler is an optional add-on to IBM Cloud Orchestrator and it is not shipped as part of IBM Cloud Orchestrator but it must be obtained through IBM separately.

To use Platform Resource Scheduler with IBM Cloud Orchestrator, first install IBM Cloud Orchestrator, then contact IBM to obtain and install Platform...
Resource Scheduler with a 90 day evaluation license. To use Platform Resource Scheduler in production, contact IBM to purchase a full license.

To install Platform Resource Scheduler, use the Platform Resource Scheduler installation instructions.

Once installed, you can learn more about Platform Resource Scheduler and start to use it.

Deployment topologies

Before you start to deploy IBM Cloud Orchestrator, you must decide which deployment topology to install for the IBM Cloud Orchestrator management stack.

IBM Cloud Orchestrator supports the following deployment topologies:
- Demo topology
- Distributed (not Active-Active) topology
- Distributed Active-Active (High Availability) topology

If you have IBM Cloud Orchestrator Enterprise Edition, you can install additional components such as IBM Tivoli Monitoring and SmartCloud Cost Management. For more information, see "IBM Cloud Orchestrator Enterprise Edition installation" on page 15.

The first step for each deployment topology is to install the Deployment Service on a dedicated system. The descriptions of the IBM Cloud Orchestrator topologies below do not mention the Deployment Service. The topology descriptions in this topic focus only on the specific IBM Cloud Orchestrator components.

To install the required deployment topology, use the appropriate deployment template. For more information about templates, see "Understanding deployment templates" on page 38.

Demo topology

This topology is the simplest topology, and it is suitable for demo and proof-of-concept scenarios. It requires minimal resources, and does not use Neutron networking or high-availability features. It supports a single VMware or KVM region. It does not support more than one single region.
The topology involves two systems:

- The first system acts as both Central Server and Region Server. It hosts the IBM HTTP Server, Business Process Manager, the Self-service user interface, the Administration user interface, all the OpenStack components, qpid, and the Public Cloud Gateway. You can install the database on this system or you can use an external database.
- The second system hosts the Workload Deployer component.

**Distributed (not Active-Active) topology**

This topology is typically used for test environments in which you can use all the product features, but do not want to invest in resources for high availability. IBM Cloud Orchestrator components are installed on three Central Servers. It supports one or more Region Servers and it supports one or more OpenStack Neutron servers.

Central Server 1 hosts the database and OpenStack Ceilometer. You can install the database on this system or you can use an external database.

Central Server 2 hosts the IBM HTTP Server, Business Process Manager, the Public Cloud Gateway, the Self-service user interface, the Administration user interface, and OpenStack Keystone.

Central Server 3 hosts the Workload Deployer component.

Each region server hosts OpenStack Nova, Glance, Cinder, Heat, and qpid. If you want to use OpenStack Neutron features, you must install OpenStack Neutron on a dedicated system.

**Note:** The keystone CLI is always installed on the same system where OpenStack Keystone is installed. All of the other OpenStack CLI commands are available on the Region Server.

**Distributed Active-Active (High Availability) topology**

This topology is the most advanced topology for the IBM Cloud Orchestrator management stack. It provides improved availability of the IBM Cloud Orchestrator management stack by removing single points of failure (SPOF) for various components. This improvement is achieved by application clustering (for example, Business Process Manager cluster, Keystone). Also, on certain systems, a high-availability cluster solution is automatically installed and configured to provide automated failover of failed components (for example, high-availability clustering for the haproxy load balancer). Furthermore, System Automation
Application Manager is installed on a separate server to automatically restart failed components. System Automation Application Manager also improves startup and shutdown times for the IBM Cloud Orchestrator management stack by running the start or stop sequences in parallel.

For the Distributed Active-Active (High Availability) topology, you must install an external database as described in “Configuring an external database” on page 48. It is advisable to set up an IBM DB2 database with a high-availability solution because the database is a core component of the IBM Cloud Orchestrator management stack.

The active-active Central Server setup consists of four systems:
- Central Server 1 hosts OpenStack Ceilometer.
- Central Server 2 hosts the IBM HTTP Server, Business Process Manager, haproxy, the Public Cloud Gateway, the Self-service user interface, the Administration user interface, Tivoli System Automation, and OpenStack Keystone.
• Central Server 2 hosts the IBM HTTP Server, Business Process Manager, haproxy, the Self-service user interface, the Administration user interface, Tivoli System Automation, and OpenStack Keystone.
• Central Server 3 hosts the Workload Deployer component.

An additional system is needed to run System Automation Application Manager.

The active-active Region Server setup consists of two or three systems:
• Two systems for the active-active setup of the core region components (for example, nova-api, glance-api, and others). Each region server hosts OpenStack Nova, Glance, Cinder, Heat, and qpid.
• If you want to use OpenStack Neutron features, you must install OpenStack Neutron on a dedicated system. The Neutron system is not set up in active-active mode.

To make this implementation resistant to hardware failures, you can integrate the high-availability features provided by System Automation Application Manager and Tivoli System Automation with VMware high-availability features.

**Note:** The keystone CLI is always installed on the same system where OpenStack Keystone is installed. All of the other OpenStack CLI commands are available on the Region Server.

**Hardware prerequisites**

The hardware prerequisites are based on the role of the node you are using to deploy the IBM Cloud Orchestrator.

<table>
<thead>
<tr>
<th>Machine Role</th>
<th>Processor (vCPU)</th>
<th>Memory (GB)</th>
<th>Overall free volume requirements (GB)</th>
<th>Free space in directories (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>/</td>
<td>/home</td>
</tr>
<tr>
<td>Deployment Server</td>
<td>1</td>
<td>4</td>
<td>117</td>
<td>15</td>
</tr>
<tr>
<td>Central Server 1</td>
<td>2</td>
<td>6</td>
<td>100</td>
<td>75</td>
</tr>
<tr>
<td>Central Server 2</td>
<td>2</td>
<td>8</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>Central Server 3</td>
<td>2</td>
<td>6</td>
<td>146</td>
<td>77</td>
</tr>
<tr>
<td>KVM Region Server</td>
<td>2</td>
<td>4</td>
<td>77</td>
<td>40</td>
</tr>
<tr>
<td>KVM Compute Node</td>
<td>4</td>
<td>32</td>
<td>160</td>
<td>80</td>
</tr>
<tr>
<td>VMware Region Server</td>
<td>2</td>
<td>8</td>
<td>77</td>
<td>40</td>
</tr>
<tr>
<td>Power® Region Server</td>
<td>2</td>
<td>4</td>
<td>77</td>
<td>40</td>
</tr>
<tr>
<td>z/VM® Region Server</td>
<td>2</td>
<td>4</td>
<td>77</td>
<td>40</td>
</tr>
</tbody>
</table>
**Table 2. The hardware prerequisites (continued)**

<table>
<thead>
<tr>
<th>Machine Role</th>
<th>Processor (vCPU)</th>
<th>Memory (GB)</th>
<th>Overall free volume requirements (GB)</th>
<th>Free space in directories (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutron Network Server</td>
<td>2</td>
<td>4</td>
<td>32</td>
<td>/</td>
</tr>
<tr>
<td>Server for all-in-one deployment with KVM Region</td>
<td>4</td>
<td>16</td>
<td>350</td>
<td>/</td>
</tr>
<tr>
<td>Server for all-in-one deployment with VMware Region</td>
<td>4</td>
<td>16</td>
<td>377</td>
<td>/</td>
</tr>
<tr>
<td>Server for System Automation Application Manager</td>
<td>4</td>
<td>8</td>
<td>20</td>
<td>/</td>
</tr>
</tbody>
</table>

Disk planning considerations:

- The specified hard disk space is the minimum free space required on the machine before the IBM Cloud Orchestrator installation. Be sure that there is sufficient space for the required partitions.
- For the Deployment Service node, an additional disk size (about 14 GB) is required to store the IBM Cloud Orchestrator packages. If you plan to install IBM Cloud Orchestrator as highly available, you need 40 GB to store all of the IBM Cloud Orchestrator packages, including the additional high-availability packages.
- For Central Server 1, an additional space on `/home` may be required over time based on database size. Monitoring is recommended.
- It is recommended to use LVM to manage the partition so that you can extend the size if needed.
- For Central Server 3, the `/drouter` directory is used to store the contents (images and patterns) of the Workload Deployer component. Increase the partition size according to the contents you want to deploy.
- For KVM Compute node, because the virtual machine master images and the virtual machine ephemeral disks are located in the `/var/lib/nova` directory by default, plan the required disk space for this directory accordingly.

For demo topology, in addition to the server for all-in-one deployment, only one central server is needed. This central server, where the Workload Deployer component is installed, must satisfy the Central Server 3 hardware requirements. For more information about the demo topology, see "Deployment topologies" on page 16.

For more information about hardware requirements for the System Automation Application Manager server, see the Tivoli System Automation Application

For Neutron Network Node and KVM Compute Node, it is recommended to have physical machines for better performance.

A different Neutron Network Node is required for each Region Server that uses Neutron network.

For IBM Cloud Orchestrator environment with high availability configuration, you need a secondary Central Server 2 and a secondary Region Server (for each primary Region Server in your environment) with the same prerequisites specified in the previous table. These secondary virtual servers should reside on another host than the primary virtual servers, to really cover the outage of the primary host.

The following IBM Power processor-based servers are supported:
- IBM POWER6®
- POWER6+™
- POWER7®
- POWER7+™
- POWER8™

IBM Flex System Power Architecture compute nodes. Managed hosts must have a minimum of 4 cores and 8 GB of memory. All Power hosts using SAN storage must have Fibre Channel cards that are suitable for connecting to the SAN switches.

**Software prerequisites**

Review the software prerequisites for your environment.

IBM Cloud Orchestrator runs on top of Red Hat Enterprise Linux.

The installer needs to access one of the following Red Hat Enterprise Linux repositories:
- Registered Red Hat Network (RHN)
- Customer-provided yum repository.
- Red Hat Enterprise Linux ISO

**Note:**
- Before you install the product, ensure that you have updated the RHEL image by using yum repositories or RHN. The yum repositories should contain os, update and load balancer repository. For information about heartbleed vulnerability, see [https://access.redhat.com/solutions/781793](https://access.redhat.com/solutions/781793).
- To enable the load balancer repository for a specific system managed by RHN, select the system from the **Systems** tab of the RHN interface, and make sure that **RHEL Server Load Balancer channel** is checked in the Software Channel Subscriptions list (**Software > Software Channels**).

**Manage-from requirements**

IBM Cloud Orchestrator services can be installed on KVM or VMware virtual machines, or on physical machines.
Depending on the deployment scenario that you choose, the Deployment Service uses already existing virtual images or it creates them and then installs the software stack on them.

If the Deployment Server is in charge of creating the virtual machines, the virtualization technology (VT) must be enabled on the operating system running the Deployment Service.

The following table describes the host and guest operating systems supported for installing IBM Cloud Orchestrator.

### Table 3. Host and guest operating systems supported by the standard installation

<table>
<thead>
<tr>
<th>Hypervisor</th>
<th>Host operating system</th>
<th>Guest operating system</th>
<th>Reference</th>
</tr>
</thead>
</table>
| KVM        | Red Hat Enterprise Linux 6.4 or 6.5, x86_64 | Red Hat Enterprise Linux 6.4 or 6.5, x86_64 | [Managing guests with the Virtual Machine Manager](#)  
[virt-manager](#) |
| VMware     | VMware vCenter Server 5.0, 5.1, 5.1 u2, 5.5, or 5.5 u1  
VMware vSphere 5.0, 5.1, 5.1 u2, 5.5, or 5.5 u1 | VMware vCenter Server 5.0, 5.1, 5.1 u2, 5.5, or 5.5 u1 | [vSphere Virtual Machine Administration](#) |

**Note:**
- Be sure that all the IBM Cloud Orchestrator central servers run the same Red Hat Enterprise Linux version.
- If you want to use OpenStack neutron, the server that hosts the neutron service must run Red Hat Enterprise Linux 6.5 and iproute must be upgraded to the version 2.6.32-130. The `iproute-2.6.32-130.el6ost.netns.2.x86_64.rpm` package can be downloaded from [http://repos.fedorapeople.org/repos/openstack/openstack-icehouse/epel-6/](http://repos.fedorapeople.org/repos/openstack/openstack-icehouse/epel-6/)
- If you plan to use an existing external database, Red Hat Enterprise Linux 6.4 or 6.5 64 bits must be installed on the database system and you must use DB2 10.5.
- If you migrated from SmartCloud Orchestrator 2.3 and you were using VMware 4.1, you must upgrade to VMware 5.0 or later before using IBM Cloud Orchestrator 2.4.

You must apply the following configurations to the IBM Cloud Orchestrator servers:

- The operating systems must be installed at least with the basic server package group.
- The `bind-utils` rpm must be installed.
- The SSH daemon must be enabled.
- The network interface must be configured to use static IPs.
- All the IBM Cloud Orchestrator servers must be in the same network.
- All the IBM Cloud Orchestrator servers must have network connectivity.
- The network interface used as management network (for example, the network used to connect with the IBM Cloud Orchestrator servers) must have the same name on all the IBM Cloud Orchestrator servers (for example, eth0).
- Host name resolution must work across all of the IBM Cloud Orchestrator servers. You can configure the IBM Cloud Orchestrator servers with the corporate DNS. If no corporate DNS is available, you must update the `/etc/hosts` file on each of the required IBM Cloud Orchestrator servers (for example, Central Servers, Region Servers, compute nodes) to include all of the
IBM Cloud Orchestrator server hosts. Each entry in the /etc/hosts file should specify both the fully qualified domain name and the host name, in that order. To verify that you configured the /etc/hosts file correctly, run the following commands:

```
hostname --fqdn
```

This command must return the FQDN of the server (for example, `central_server_2.subdomain.example.com`).

```
hostname
```

This command must return the first part of the FQDN, that is the host name (for example, `central_server_2`).

### Manage-to requirements

The following table describes the host and guest operating systems supported by each type of hypervisor in a IBM Cloud Orchestrator environment.

<table>
<thead>
<tr>
<th>Hypervisor</th>
<th>Host operating system</th>
<th>Guest operating systems of deployed virtual machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>KVM</td>
<td>Red Hat Enterprise Linux 6.4 or 6.5, x86_64</td>
<td>SUSE Linux Enterprise Server 11 SP1, SP2, or SP3, 32-bit or 64-bit</td>
</tr>
<tr>
<td>VMware</td>
<td>VMware vCenter Server 5.0, 5.1, 5.1 u2, 5.5, or 5.5 u1 VMware vSphere 5.0, 5.1, 5.1 u2, 5.5, or 5.5 u1</td>
<td>Red Hat Enterprise Linux 5.x, Red Hat Enterprise Linux 6.1, 6.2, 6.3, 6.4, or 6.5, 32-bit or 64-bit CentOS 6.3 or 6.4, 64-bit Windows 2008 R2 64-bit, Windows 7 64-bit, Windows Server 2012 64-bit, Windows 8 64-bit</td>
</tr>
<tr>
<td>PowerVC</td>
<td>PowerVC 1.2.1 Fix Pack 2 HMC 7.8.1 VIOS 2.2.3 HMC 8.1.0 VIOS 2.2.3.3 for Shared Storage Pool support</td>
<td>AIX 6.1 TL9 or later, AIX 7.1 TL3 or later Red Hat Enterprise Linux 5.9 or later, Red Hat Enterprise Linux 6.4 or later, Red Hat Enterprise Linux 7.0 or later SUSE Linux Enterprise Server 11 SP3 or later</td>
</tr>
<tr>
<td>z/VM</td>
<td>z/VM 6.3</td>
<td>Red Hat Enterprise Linux 6.3, 6.4, or 6.5 SUSE Linux Enterprise Server 11.1</td>
</tr>
</tbody>
</table>

**Note:**

- If you migrated from SmartCloud Orchestrator 2.3 and you were using VMware 4.1, you must upgrade to VMware 5.0 or later before using IBM Cloud Orchestrator 2.4.
- To allow IBM Cloud Orchestrator to connect to VMware vCenter, PowerVC, or z/VM, you must use an account with specific privileges. For VMware vCenter, the list of minimum permissions is available at [http://docs.openstack.org/trunk/config-reference/content/vmware.html](http://docs.openstack.org/trunk/config-reference/content/vmware.html).
- For z/VM 6.3 systems, you must install the PTFs listed at:
- z/VM Single System Image (SSI) configuration is not supported in IBM Cloud Orchestrator V2.4.
Planning networks

IBM Cloud Orchestrator supports Nova networks and Neutron networks.

You can use Nova networks and Neutron networks in one or more different regions, but you cannot use Nova networks and Neutron networks within the same region.

In PowerVC and z/VM regions, you can only use Neutron networks.

Nova networks do not support overlapping IP addresses.

Neutron provides rich and flexible scenarios for end users compared to Nova network. For example, it supports software defined networks (SDN) via Virtual Extensible LAN (VXLAN) and it also supports traditional routing accessibility such as VLAN and FLAT.

Note: You cannot use VXLAN for VMware managed-to environment.

Prerequisites for Nova networks

- The switch must be configured as trunked to support different VLAN IDs.
- There must be a routable gateway for each VLAN on the router.
- The IP address of the gateway must belong to the subnet that you plan to exploit.
- If using KVM regions, all the Compute Nodes must use the same interface to connect to the Region Server (for example, all eth0 or all eth1).
- If the Region Server is a VMware virtual machine, the port group must have the VLAN ID option set to All (4095).
- VMware Region Servers must be in the same network as all the ESXi servers.
- KVM Region Servers can be in a different network than the Compute Nodes.

Prerequisites for Neutron networks

- The Region Server can be located in a different network than the Compute Nodes.
- The Neutron server must be in the same network of the Compute Nodes.
- In a VMware region, the port groups are not created by the installer. The name of the port group must match the label of the network in Neutron.

Using a virtual machine as network server

You can install the network server (either Neutron or Nova) on a virtual machine.

The network node must be in the same network as the compute nodes. The same vNICs must correlate to the same NICs on the compute nodes, as shown in the following topology diagram.
where:

**MGM Hyper**
- The management hypervisor, which can be either Linux KVM or VMware ESXi

**Network Node**
- The Neutron node if you are using Neutron networks, or the Region Server if you are using Nova networks.

**Compute Node**
- The compute node where the virtual machines are deployed.

**Br** Bridge

**Pg** Port group

**eth0/vmnic0**
- Examples of NIC names

The following prerequisites must be met:

**For a VMware virtual machine:**
- The port groups to which the network server connects must have the VLAN ID option set to All (4095).
- The port groups to which the network server connects must be configured to accept Promiscuous Mode.
- All the vNICs of the corresponding port groups must correlate correctly to the NICs of the compute nodes. For example, the connected port group eth0 of the network server must be in the same network as the eth0 of the compute nodes, the connected port group eth1 of the network server must be in the same network as the eth1 of the compute nodes, and so on.
- The network adapter of the vNIC on the network node must be 'E1000'. If you use the 'VMXNET3', you may hit the problem that UDP packets are dropped from the network node.

**For a KVM virtual machine:**
- The virtual machine must not be running on a compute node.
- All the vNICs of the corresponding bridges must correlate correctly to the NICs of the compute nodes. For example, the connected bridge eth0 of the network server must be in the same network as the eth0 of the compute nodes, the connected bridge eth1 of the network server must be in the same network as the eth1 of the compute nodes, and so on.
The following limitations apply when you install a network server on a KVM virtual machine:

**VXLAN**

The network server must communicate with other compute nodes by using multicast to transmit overlay packets. If the network server is running as a virtual machine, it relies on the bridge of its hypervisors to connect externally. A known issue for the kernel of Red Hat Enterprise Linux V6.5 is related to multicast snooping for bridges: sometimes the multicast packets cannot be received by the virtual machine. If you see this problem, run the following command to disable multicast snooping in the kernel:

```
echo 0 > /sys/devices/virtual/net/BRIDGE-OF-NETWORK-SERVER/bridge/multicast_snooping
```

where **BRIDGE-OF-NETWORK-SERVER** is the bridge of the data interface of the network server.

**VLAN**

All 8021q tagged packets must be transmitted transparently through the bridge in the hypervisor. If you create a VLAN on the network server with Neutron or Nova network, you must ensure that the same VLAN devices are not used on the hypervisor. For example: you create a network with VLAN 100 with physnet1 (eth0) on the network server. In this case, on the hypervisor of the network server, do not create VLAN device eth0.100. You can create VLAN device eth0.100 on other VLANs that are not on the network server. If you really need the same VLAN to work on the hypervisor, create a VLAN device based on brx instead of ethx. For example, use the following command to create br0.100:

```
vconfig add br0 100
```

Summary of features and capabilities of a network server:

This table summarizes the features and capabilities of deploying a network server on a virtual machine or on a physical server.

| Table 5. Comparison of network types and hypervisors to deploy network servers |
|---------------------------------|---------------------------------|-----------------|-----------------|
| FLAT                            | FLAT DHCP                       | VLAN            | VXLAN           |
| Nova-network on VMware virtual machine | No (yes only for systems upgraded from SmartCloud Orchestrator) | No              | Yes             | No              |
| Neutron on VMware virtual machine  | Yes                             | Not applicable  | Yes             | Yes             |
| Nova-network on KVM virtual machine | No (yes only for systems upgraded from SmartCloud Orchestrator) | No              | Yes             | No              |
| Neutron on KVM virtual machine   | Yes                             | Not applicable  | Yes             | Yes             |
| Nova-network on physical server  | No (yes only for systems upgraded from SmartCloud Orchestrator) | No              | Yes             | No              |
Table 5. Comparison of network types and hypervisors to deploy network servers (continued)

<table>
<thead>
<tr>
<th></th>
<th>FLAT</th>
<th>FLATDHCP</th>
<th>VLAN</th>
<th>VXLAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutron on physical server</td>
<td>Yes</td>
<td>Not applicable</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

For information about limitations, see “Using a virtual machine as network server” on page 24.

**Neutron scenarios**

Learn the network scenarios supported in Neutron networks.

Neutron supports the following scenarios:

**Network Server as a gateway**

This scenario is suitable when SDN is required and one of the important benefits is that network fencing can be achieved as well. In this scenario, virtual machines get their internal IP addresses from the DHCP service and access internet through the SDN gateway. Because the virtual machines have only internal IP addresses, the only way to access them is through floating IP addresses. So, network server provides:

- DHCP services for different networks.
- A gateway for different networks.
- Floating IP addresses to the virtual machines to be accessed.

**Note:**

- This scenario can be exploited by using either VLANs or VXLANs.
- Multiple routers are allowed for different networks.

A Cloud Service Provider (CSP) is one type of customer that can leverage this Neutron configuration. A CSP usually has to support multiple groups of end users (tenants). Beyond providing network isolation for the resources provided to each tenant, a CSP may also need to provide to some tenants the ability to access these resources from the public internet. Leveraging the configuration described in this scenario, the CSP is able to provide private resources to its tenants when they connect to the environment via, for example, a VPN connection. This configuration also allows the CSP to provide internet accessible resources directly to tenants and their users. An example of the private resources can be a development or test environment, where a tenant develops a new application, that should not be accessible directly via the internet. An example of an internet accessible resource could be an HTTP server hosting the development project metrics that can be shared with externally interested parties.

**Physical routing device as a gateway**

This scenario is quite similar to what Nova network provides, that is, you must plan carefully the overall topology because all the virtual machines must be able to be accessed by using their internal IP addresses, so VLAN trunks and gateway must be configured properly on switch and router. Differently from the first scenario, the network server in this deployment provides only DHCP services, you cannot use any routing service or floating IP, and network fencing (overlapping IP) is not supported. Choose this scenario when traditional network for the virtual machines is more suitable.
Note: Besides VLAN, you can also use FLAT network.
A user who is looking to create a private cloud leveraging an existing virtual environments can use the configurations described in this scenario. For example, an IT Operations team trying to provide better access, improved SLAs and to satisfy more dynamic compute demands from different internal teams can use this configuration. Environments that this simpler configuration supports usually provides network isolation via the use of VLANs and subnetworks for the different tenants (usually internal teams). In most cases, there is no need for these resources to be externally accessible nor does the environment employ extensive use of Network Address Translation (NAT) and overlapping IP addresses.

Hybrid SDN plus physical routing
This is a combination of the first and second scenarios that can be deployed in the same environment. Network servers work as SDN gateway for some of the networks, and for other networks, we use physical devices to achieve the routing accessibility. Therefore, you can deploy complex network scenarios accordingly.

Planning a Distributed Active-Active (High Availability) installation

Ensure that your environment meets the requirements for a high-availability installation, to provide enhanced high availability of the IBM Cloud Orchestrator management stack.

Before you start the installation, review the requirements and plan the entire process. Remember that a high-availability installation requires more hardware than an installation that is not highly available, and additional process steps are needed.

High-availability hardware requirements

To install IBM Cloud Orchestrator, you need the hardware that is listed in "Hardware prerequisites" on page 19. For a high-availability installation, you need the following additional nodes for redundancy:
- Secondary Central Server 2
- Secondary KVM Region Server
- Secondary VMware Region Server

Important: Create these secondary virtual servers on a different physical host than the primary virtual servers, to ensure that the IBM Cloud Orchestrator management stack remains available if the primary host fails.

Restriction: For a high-availability installation, only VMware and KVM regions are supported. IBM Cloud Orchestrator V2.4 does not support the use of a high-availability Central Server installation with a PowerVC region or z/VM region.

External database

For a high-availability installation, you must use an external database, as described in "Configuring an external database" on page 48.
Preparing the installation media

Before you install, prepare the installation media.

Procedure
1. If you want to install IBM Cloud Orchestrator, download the related images from the IBM Passport Advantage® site. The image file names are:
   ICO_V240_1of4.tar
   ICO_V240_2of4.tar
   ICO_V240_3of4.tar
   ICO_V240_4of4.tar

   If you want to install IBM Cloud Orchestrator Enterprise Edition, download the related images from the IBM Passport Advantage site. The image file names are:
   ICO_Ent_V240_1of4.tar
   ICO_V240_2of4.tar
   ICO_V240_3of4.tar
   ICO_V240_4of4.tar

2. Extract the image files to Deployment-Service node disk to the /opt/ico_install directory, for example, by using the following command for each image file:
   tar -xf <image_file>.tar -C /opt/ico_install

3. Download the following Business Process Manager packages and copy them to the /opt/ico_install/data/orchestrator-chef-repo/packages/bpm_binaries/ directory:
   BPM_Std_V85_Linux_x86_1_of_3.tar.gz
   BPM_Std_V85_Linux_x86_2_of_3.tar.gz
   BPM_Std_V85_Linux_x86_3_of_3.tar.gz

4. Download the following IBM HTTP Server packages and copy them to the /opt/ico_install/data/orchestrator-chef-repo/packages/ihs_binaries/ directory:
   WAS_V85_SUPPL_3_OF_3.zip
   WAS_V85_SUPPL_2_OF_3.zip
   WAS_V85_SUPPL_1_OF_3.zip

5. If you want to install IBM Cloud Orchestrator with high availability capabilities, the following packages are also needed. Otherwise, set package_checking="no" in /opt/ico_install/installer/deployment-service.cfg, to allow the installer to proceed without these packages.
   a. Download the following System Automation for Multiplatforms package and copy it to the /opt/ico_install/data/orchestrator-chef-repo/packages/samp/ directory:
      SA_MP_4.1_Linux.tar
   b. Download the following System Automation Application Manager package and copy it to the /opt/ico_install/data/orchestrator-chef-repo/packages/saam/ directory:
      SA_AM_4.1_LinSysX.tar
   c. Download the following Jazz for Service Management package and copy it to the /opt/ico_install/data/orchestrator-chef-repo/packages/jazzsm directory:
      JAZZ_FOR_SM_1.1.0.2_FOR_LINUX_ML.zip
   d. Download the following WebSphere® Application Server for Jazz for Service Management package and copy it to the /opt/ico_install/data/orchestrator-chef-repo/packages/was directory:
e. Download the following IBM DB2 package and copy it to the
   /opt/ico_install/data/orchestrator-chef-repo/packages/db2 directory:
   IBM_DB2_10.1_-_Limited_Use_for_Li.tar.gz

6. If you are using RHEL ISO file instead of Red Hat Subscription Management or
   yum as package repository, copy the ISO file to the /opt directory on the
   Deployment Service node.

7. If you want to install the additional components for IBM Cloud Orchestrator
   Enterprise Edition, see the Download Document for a list of the images to be
   downloaded for:
   • IBM SmartCloud Cost Management
   • IBM Tivoli Monitoring
   • IBM Tivoli Monitoring for Virtual Environments
   • Jazz for Service Management

   For information about installing IBM Cloud Orchestrator Enterprise Edition, see
   “Preparing to install additional Enterprise Edition components” on page 93.

**Preparing resources to install IBM Cloud Orchestrator**

To start the installation process, you must prepare several virtual machine or
physical machine to install the IBM Cloud Orchestrator.

**Before you begin**

Depending on the scenarios, you have several choices to make regarding the
topology and the configuration of your environment. You can deploy multiple
environments for different purpose.

You can use VMware or KVM virtual machines to run the IBM Cloud Orchestrator
management from servers. To manage KVM virtual machines, it is recommended
that you use the virt-manager application. To manage VMware virtual machines, it
is recommended that you use the VMware vCenter application. If you want to use
the high availability solution in your IBM Cloud Orchestrator environment, use
VMware virtual machines to run your IBM Cloud Orchestrator environment.

If you plan to exploit the high availability features, use VMware virtual machines.
For more information about high availability, see “High-availability solutions” on
page 187.

If the virtual machines are already available or you plan to use physical machines,
use the following procedure. If you need to create the virtual machines, use the
procedure described in “Setting up virtual machines” on page 32.

**About this task**

Repeat the described procedure for each of the virtual machines to prepare them
for the installation process.

**Procedure**

1. Make sure that the virtual machines meet the hardware prerequisites described
   in “Hardware prerequisites” on page 19
2. Install Red Hat Enterprise Linux V6.4 or V6.5 64-bit on the virtual machines:
Make sure that all virtual machines have the same kernel version installed and that they are consistent with the YUM repository or the ISO file that you prepared.

When installing the Red Hat Enterprise Linux V6.4 or V6.5 64-bit operating system for the IBM Cloud Orchestrator servers, the Basic Server package group is sufficient, because the IBM Cloud Orchestrator servers deploy script installs the required packages from the corresponding YUM repository or Red Hat ISO files.

**Note:** Make sure that the Red Hat Enterprise Linux installation includes the bind-utils RPM.

3. Verify that the SSH daemon is enabled on the virtual machine.

4. Make sure that the network is configured for the virtual machine, preferably the static IP. All IBM Cloud Orchestrator server virtual machines must:
   - Be in the same network
   - Have network connectivity
   - Have the same NIC name (such as eth1/eth0.309) for the management network used to manage IBM Cloud Orchestrator

5. Configure the Domain Name Server (DNS). Configure the IBM Cloud Orchestrator servers with the Corporate DNS, if there is no Corporate DNS available, use /etc/hosts to have all the IBM Cloud Orchestrator servers hosts in it. Configure the Fully Qualified Domain Name (FQDN) for each IBM Cloud Orchestrator required server, to verify that you can run it.

   host <IP_address>
   This command must return the FQDN of the server.

   hostname --fqdn
   This command must return the same FQDN as in the previous command.

   hostname
   This command must return the first part of the FQDN.

**Note:**
   - You can specify the FQDN for creating a node, if no Corporate DNS is in use and there is no host mapping in /etc/hosts.
   - Put FQDN in order of priority. Corporate DNS, hosts' mapping is in /etc/hosts. Specify it when you create the node.

6. Make sure that SELINUX is disabled on Central Server 2. To disable SELINUX, edit the /etc/selinux/config file and set the parameter SELINUX=disabled. Reboot the node.

7. Remove the following packages on the Deployment Service node and on each IBM Cloud Orchestrator server, if installed:
   - puppet
   - facter
   - hiera
   - mcollective
   - mcollective-common
   - subversion-ruby
   - libselinux-ruby
   - ruby-augeas
   - ruby-rgen
   - ruby-shadow
   - rubygem-json
   - rubygem-stomp
   - rubygems
Run the following commands:

```bash
yum -y remove puppet, facter, hiera, mcollective, mcollective-common, subversion-ruby,
libselinux-ruby, ruby-augeas, ruby-rgen, ruby-shadow, rubygem-json,
rubygem-stomp, rubygems

yum -y remove puppetlabs-release-6-10.noarch
```

**Setting up virtual machines**

If you want to use virtual machines to install IBM Cloud Orchestrator and you do not want to have the Deployment Server to create them, follow this procedure.

**Setting up KVM virtual machines**

To set up a KVM virtual machine, install and configure the KVM-related software on the host system and create the virtual machines as described in the following procedures. It is recommended to use the same operating system version on all the virtual machines you use for installing IBM Cloud Orchestrator.

**Prerequisites**

- The host machines must use Intel VT or AMD-V chipset supporting hardware-assisted virtualization.
- Virtualization support must be enabled in the BIOS of your host machine.
- Verify that the processor supports KVM by running the following command on your host system:
  ```bash
grep -E 'vmx|svm' /proc/cpuinfo
  ```
  If the command returns output, your system supports KVM.
- The following packages must be installed on the host:
  - `kvm`
  - `virt-manager`
  - `libvirt`
  - `libvirt-python`
  - `python-virtinst`
  - `tunctl`

If the packages are not installed, you can install them by using a yum repository.

To run the following procedure, you must have access to the RHEL ISO, but you can exploit different package repositories.

1. Configure an ISO repository to be available to your host for downloading the KVM-related packages. To create a repository using an ISO image, perform the following steps:
   a. For example, if you have your RHEL 6.x ISO image stored in `/opt/RHEL6.x-xxxxxxx.x-Server-x86_64-DVD1.iso`, mount the ISO image by running the following commands:
      ```bash
      mkdir -p /mnt/rhel6
      mount -o loop /opt/RHEL6.x-xxxxxxx.x-Server-x86_64-DVD1.iso /mnt/rhel6
      ```
   b. Create a repo file in the `/etc/yum.repos.d` directory and change the base path to the directory path that you specified in the mount command. For example, create the `/etc/yum.repos.d/DVD.repo` file with the following content:
2. Install the KVM management packages via yum by running the following command:

   yum install kvm virt-manager libvirt libvirt-python python-virtinst tunctl

Operating system configuration

Before creating the virtual machine, you must apply specific configuration changes to the host operating system. Run the following procedure using root privileges:

1. Disable NetworkManager by running the following commands:
   - chkconfig NetworkManager off
   - chkconfig network on
   - service NetworkManager stop
   - service network start

2. Start the libvirtd service by running the following command:
   - service libvirtd start

3. Create a bridge (br0) based on the eth0 interface by running the following command on the host:
   - virsh iface-bridge eth0 br0

   If the command fails, you can manually configure the networks by performing the following steps:
   a. Edit your /etc/sysconfig/network-scripts/ifcfg-eth0 file as in the following example:
      
      ```
      DEVICE=eth0
      ONBOOT=yes
      BRIDGE=br0
      ```
      
   b. Edit your /etc/sysconfig/network-scripts/ifcfg-br0 file as in the following example:
      
      ```
      DEVICE=br0
      ONBOOT=yes
      TYPE=Bridge
      BOOTPROTO=none
      IPADDR=192.168.1.2
      NETMASK=255.255.255.0
      GATEWAY=192.168.1.1
      STP=on
      DELAY=0
      ```
      
   c. Restart the network by running the following command:
      - service network restart
      
   d. You can review your interfaces and bridge by running the following commands:
      - ifconfig
      - brctl show

Creating the virtual machines

You can run one of the following procedures to create a KVM virtual machine. If you are not an experienced KVM user, it is recommended to use the first procedure:

- Use virt-manager
The virt-manager application is a desktop user interface for managing KVM virtual machines. It uses a GUI based front end for KVM creation, installation and monitoring. It also allows an easy method for adding hardware. Launch virt-manager and follow the wizard.

- **Use the command line**

Perform the following steps:

1. Create a base qcow2 image for Central Server and Region Server by running the following command:

   `qemu-img create -f qcow2 -o cluster_size=2048k /home/sco-base.qcow2 200G`

2. Creating guests with virt-install, by running the following command:

   ```
   virt-install --connect=qemu:///system
   --name=sco-base
   --ram=2048
   --vcpus=2
   --arch=x86_64
   --os-type=linux
   --os-variant=rhel6
   --hvm
   --virt-type kvm
   --cdrom=/opt/RHEL6.x-xxxxxxxx.x-Server-x86_64-DVD1.iso
   --disk path=/home/sco-base.qcow2,format=qcow2,bus=virtio
   --network bridge=br0,model=virtio
   --accelerate
   --vnc
   ```

   where

   **--arch** Specifies the bits of the guest operating system. Use x86_64 for IBM Cloud Orchestrator virtual machines.

   **--os-variant**

   Specifies the detailed variant of the guest operating system. You can get the full list by running the `man virt-install` command.

   **--disk**

   Specifies the virtual disk or disks for installing the guest operating system. If supported, enable virtio.

   **--network**

   Specifies the bridge device name. If supported, enable virtio.

   The virt-viewer window is displayed. Follow the steps in the installation wizard to complete the RHEL installation. Install the operating system with base mode. For operating systems that support virtio, customize the partition layout with only `/filesystem`.

3. Configure the image file by performing the following steps:

   a. Log in to the operating system by using virt-viewer:

      ```
      virtviewer <ID or name>
      ```

      For example:

      ```
      virtviewer sco-base
      ```

      Run the `virsh list --all` command to get a list of all the IDs or names.

   b. Disable SELinux by editing the `/etc/selinux/config` file and specifying the following value:

      ```
      SELINUX=disabled
      ```

   c. Disable the firewall by running the following commands:
service iptables stop
service ip6tables stop
chkconfig iptables off
chkconfig ip6tables off
d. Edit the /etc/hosts file and specify the following value:
   # Do not remove the following line, or various programs
   # that require network functionality will fail.
   127.0.0.1 localhost.localdomain localhost
e. Edit the /etc/sysconfig/network file and specify the following value:
   NETWORKING=yes
f. Edit the /etc/sysconfig/network-scripts/ifcfg-eth0 file and specify the following values:
   # Intel Corporation 82562GT 10/100 Network Connection
   DEVICE=eth0
   BOOTPROTO=dhcp
   ONBOOT=yes
   TYPE=Ethernet
   PERSISTENT_DHCLIENT=1

Remove the write net rules by running the following commands:
   mv /lib/udev/write_net_rules /lib/udev/write_net_rules.bak
   mv /etc/udev/rules.d/70-persistent-net.rules /etc/udev/rules.d/70-persistent-net.rules.bak
   mv /lib/udev/rules.d/75-persistent-net-generator.rules
      /lib/udev/rules.d/75-persistent-net-generator.rules.bak
g. Shut down the virtual machine by running the following command:
   shutdown -h now

4. Create an image file for each server by running the following command:
   qemu-img create -b <path>/sco-base.qcow2 -f qcow2 -F qcow2 <path>/<SERVER_NAME>.qcow2
5. Create XML templates file called <SERVER_NAME>.xml for each server with the following content:

```xml
<domain type='kvm'>
  <name>SERVER_NAME</name>
  <memory unit='KiB'>SERVER_RAM</memory>
  <currentMemory unit='KiB'>SERVER_RAM</currentMemory>
  <vcpu placement='static'>CPU_NUM</vcpu>
  <os>
    <type arch='x86_64'>hvm</type>
    <boot dev='hd'/>  
  </os>
  <features>
    <acpi/>
    <apic/>
  </features>
  <clock offset='localtime'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2' cache='none'/>
      <source file='IMAGE_PATH'/>
      <target dev='vda' bus='virtio'/>
    </disk>
    <controller type='ide' index='0'/>
    <controller type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x01'/>
    <controller type='usb' index='0'/>
    <controller type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x02'/>
    <interface type='bridge'>
      <source bridge='br0'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
    </interface>
    <serial type='pty'>
```
<target port='0'/>
</serial>
<console type='pty'>
  <target type='serial' port='0'/>
</console>
<input type='tablet' bus='usb'/>
<input type='mouse' bus='ps2'/>
<graphics type='vnc' port='-1' autoport='yes'/>
<video>
  <model type='cirrus' vram='9216' heads='1'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
</video>
<memballoon model='virtio'>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</memballoon>
</devices>
</domain>

Note: To define the SERVER_RAM and CPU_NUM values for each server, see "Hardware prerequisites" on page 19. Replace SERVER_NAME and IMAGE_PATH with the actual values.

6. Create the virtual machines by using the generated XML files by running the following commands:

   virsh define <IMAGE_PATH>/<SERVER_NAME>.xml
   virsh start <SERVER_NAME>

7. Configure the virtual machines for the Deployment Server and Central Servers by performing the following steps:
   a. Run the following command to log on each virtual machine:
      
      virt-viewer <SERVER_NAME>

   b. Edit the /etc/sysconfig/network-scripts/ifcfg-eth0 file, as in the following example, if there is no DHCP server:

      # Intel Corporation 82562GT 10/100 Network Connection
      DEVICE=eth0
      BOOTPROTO=static
      ONBOOT=yes
      TYPE=Ethernet
      IPADDR=192.168.1.3
      NETMASK=255.255.255.0
      GATEWAY=192.16.1.1

   c. Update the /etc/hosts file as in the following example, if no DNS is in use:

      # Do not remove the following line, or various programs
      # that require network functionality will fail.
      127.0.0.1 localhost.localdomain localhost
      192.0.2.3 central-server1.example.com central-server1

After setting up the KVM virtual machines to install IBM Cloud Orchestrator, start the virtual machine configuration by following the procedure described in "Preparing resources to install IBM Cloud Orchestrator" on page 30.

Setting up VMware virtual machines

You can use the VMware vCenter server to build the virtual machines. For more information, see the VMware documentation.

After setting up the VMware virtual machines to install IBM Cloud Orchestrator, start the virtual machine configuration by following the procedure described in "Preparing resources to install IBM Cloud Orchestrator" on page 30.
Installing the Deployment Service

The Deployment Service is the bootstrap of the IBM Cloud Orchestrator installer. It installs all the services that are required to deploy or maintain an IBM Cloud Orchestrator environment.

Before you begin

Before you install the Deployment Service, prepare a virtual machine or a physical machine by following the procedure described in “Preparing resources to install IBM Cloud Orchestrator” on page 30 to install the operating system. When the operating system is installed and configured, copy the IBM Cloud Orchestrator package and the other required packages like Business Process Manager to the Deployment Node by following the procedure described in “Preparing the installation media” on page 29.

About this task

To install the Deployment Service, complete the following procedure. You must have already unpacked the IBM Cloud Orchestrator installation package in the /opt/ico_install directory.

Procedure

1. Customize the configuration file. Log on to the Deployment Service node, edit the /opt/ico_install/installer/deployment-service.cfg file, and modify the following items:

   - iso_location="/opt/<RHEL65 ISO file name>"
   - management_network_device="<net interface to access the other nodes>"
   - package_checking="yes"  # To ignore ICO packages checking, acceptable value is yes or no. The package checking will only print WARNING message if the value set to no, otherwise installer breaks with ERROR. Default is yes.
   - ntp_servers=""  # NTP server address/FQDN, if leave blank, installer will use Deployment Server as NTP server for whole ENVIRONMENT which deployed by ds. Please make sure the server is available. For multiple NTP servers, please use "," as division.

   Note:
   - The iso_location is optional. If this system is registered with Red Hat Subscription Management or the system is configured to use an available yum repository, the location can be blank.
   - If you are using a customized yum repository in the system, make sure that the repository configured is at the same system level as the operation system. Otherwise, the installation might fail because of package conflicts.
   - If you want to install the Deployment Service on a virtual machine, make sure the clock of the Deployment Service virtual machine is synchronized with the host clock. If there is a NTP server in your environment, you can specify the ntp_servers parameter in the deployment_service.cfg file to synchronize the clock with the NTP server that you specified.

2. Be sure that the Deployment Service node is able to resolve the host names of the other systems where you are deploying the IBM Cloud Orchestrator services.

3. Install the Deployment Service by running the following command:

   cd /opt/ico_install/installer; sudo ./deploy_deployment_service.sh
The installation log files are `/var/log/cloud-deployer/deployer_bootstrap.log` and `/var/log/cloud-deployer/deploy.log`.

It is not required to use root to install the deployment service, you can use non-root to install deployment service, and you need to make sure the user that used to install the deployment service has the `sudo` access privileges without password. To configure the user with `sudo`, you can add following line into the `/etc/sudoers`:

```
<user> ALL=(ALL)NOPASSWD:ALL
```

### Changing the Deployment Service password

After you install Deployment Service, it is recommended to change the default password of Deployment Service.

#### Procedure

1. Log in to the Deployment Service node.
2. For a root user, run the following command:
   ```
   source /root/openrc
   ```

   For a non-root user, copy the `openrc` file from `/root` to your home directory and run the following command:
   ```
   source <user_home>/openrc
   ```
3. Run the following command:
   ```
   keystone password-update
   ```
4. Create an obfuscated version of the password using `openstack-obfuscate <new password>`.
5. Modify the `/root/keystonerc` and the `/root/openrc` files with the new password that you specified in the previous step.

### Understanding deployment templates

The deployment templates define the IBM Cloud Orchestrator topology.

Depending on the installation scenario chosen, described in "Deployment topologies" on page 16, you can use the following types of templates:

#### Demo templates:

- **sco-allinone-kvm**: this template installs and configures IBM Cloud Orchestrator management stack together with a KVM-based region server with one compute node.
- **sco-allinone-VMware**: this template installs and configures IBM Cloud Orchestrator management stack with a VMware-based region server. It connects to an existing VMware vCenter. The management stack requires one or two systems (physical or virtual) depending on whether you want to install the Workload Deployer component or not.
- **sco-allinone-extdb**: this template installs and configures IBM Cloud Orchestrator management stack with an external database.
- **sco-devstack**: this template installs and configures IBM Cloud Orchestrator management stack.

#### Distributed templates:
<table>
<thead>
<tr>
<th>Template name</th>
<th>Role</th>
<th>Managed-to hypervisor</th>
<th>Network</th>
<th>Database location</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmware_region_neutron</td>
<td>VMware region server</td>
<td>VMware</td>
<td>neutron</td>
<td>on region server</td>
<td></td>
</tr>
<tr>
<td>vmware_region_neutron-sharedb</td>
<td>VMware region server</td>
<td>VMware</td>
<td>neutron</td>
<td>on central server 1</td>
<td>Cannot use sco-central-servers-extdb</td>
</tr>
<tr>
<td>vmware_region-extdb</td>
<td>VMware region server</td>
<td>VMware</td>
<td>neutron</td>
<td>external database</td>
<td></td>
</tr>
<tr>
<td>vmware_region</td>
<td>VMware region server</td>
<td>VMware</td>
<td>nova-network</td>
<td>on region server</td>
<td></td>
</tr>
<tr>
<td>vmware_region-sharedb</td>
<td>VMware region server</td>
<td>VMware</td>
<td>nova-network</td>
<td>on central server 1</td>
<td>Cannot use sco-central-servers-extdb</td>
</tr>
<tr>
<td>kvm_region-with-compute-neutron</td>
<td>KVM region server</td>
<td>KVM</td>
<td>neutron</td>
<td>on region server</td>
<td></td>
</tr>
<tr>
<td>kvm_region-with-compute-neutron-sharedb</td>
<td>KVM region server</td>
<td>KVM</td>
<td>neutron</td>
<td>on central server 1</td>
<td>Cannot use sco-central-servers-extdb</td>
</tr>
<tr>
<td>kvm_region-with-compute-extdb</td>
<td>KVM region server</td>
<td>KVM</td>
<td>neutron</td>
<td>external database</td>
<td></td>
</tr>
<tr>
<td>kvm_region-with-compute</td>
<td>KVM region server</td>
<td>KVM</td>
<td>nova-network</td>
<td>on region server</td>
<td></td>
</tr>
<tr>
<td>kvm_region-with-compute-sharedb</td>
<td>KVM region server</td>
<td>KVM</td>
<td>nova-network</td>
<td>on central server 1</td>
<td>Cannot use sco-central-servers-extdb</td>
</tr>
<tr>
<td>powervc_region_neutron</td>
<td>PowerVC region server</td>
<td>PowerVC</td>
<td>neutron</td>
<td>on region server</td>
<td></td>
</tr>
<tr>
<td>powervc_region_neutron-extdb</td>
<td>PowerVC region server</td>
<td>PowerVC</td>
<td>neutron</td>
<td>external database</td>
<td></td>
</tr>
<tr>
<td>powervc_region_neutron-sharedb</td>
<td>PowerVC region server</td>
<td>PowerVC</td>
<td>neutron</td>
<td>on central server 1</td>
<td>Cannot use sco-central-servers-extdb</td>
</tr>
<tr>
<td>zvm_region_neutron</td>
<td>z/VM region server</td>
<td>z/VM</td>
<td>neutron</td>
<td>on region server</td>
<td></td>
</tr>
<tr>
<td>zvm_region_neutron-extdb</td>
<td>z/VM region server</td>
<td>z/VM</td>
<td>neutron</td>
<td>external database</td>
<td></td>
</tr>
</tbody>
</table>
Table 6. Distributed templates (continued)

<table>
<thead>
<tr>
<th>Template name</th>
<th>Role</th>
<th>Managed-to hypervisor</th>
<th>Network</th>
<th>Database location</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>zvm_region_neutron-sharedb</td>
<td>z/VM region server</td>
<td>z/VM</td>
<td>neutron</td>
<td>on central server 1</td>
<td>Cannot use sco-central-servers-extdb</td>
</tr>
<tr>
<td>sco-central-servers</td>
<td>central servers</td>
<td>hypervisor agnostic</td>
<td>network agnostic</td>
<td>central server 1</td>
<td></td>
</tr>
<tr>
<td>sco-central-servers-extdb</td>
<td>central servers</td>
<td>hypervisor agnostic</td>
<td>network agnostic</td>
<td>external database</td>
<td></td>
</tr>
</tbody>
</table>

Distributed with high availability templates:

Table 7. Distributed with high availability templates

<table>
<thead>
<tr>
<th>Template name</th>
<th>Role</th>
<th>Managed to hypervisor</th>
<th>Network</th>
<th>Database location</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>HA-sco-central-servers-extdb</td>
<td>central servers</td>
<td>hypervisor agnostic</td>
<td>network agnostic</td>
<td>external database</td>
<td></td>
</tr>
<tr>
<td>HA-vmware_region-neutron-extdb</td>
<td>VMware region server</td>
<td>VMware</td>
<td>neutron</td>
<td>external database</td>
<td></td>
</tr>
<tr>
<td>HA-vmware_region-extdb</td>
<td>VMware region server</td>
<td>VMware</td>
<td>nova-network</td>
<td>external database</td>
<td></td>
</tr>
<tr>
<td>HA-kvm_region-with-compute-neutron-extdb</td>
<td>KVM region server</td>
<td>KVM</td>
<td>neutron</td>
<td>external database</td>
<td></td>
</tr>
<tr>
<td>HA-kvm_region-with-compute-extdb</td>
<td>KVM region server</td>
<td>KVM</td>
<td>nova-network</td>
<td>external database</td>
<td></td>
</tr>
<tr>
<td>HA-saam</td>
<td>System Automation Application Manager (SAAM) server</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can list all the available deployment templates running the ds template-list command on the Deployment Server. For each template, you can see a short description of the associated topology. For example:

source /root/keystonerc; ds template-list

```
+--------------------------------------+------------------------
| id                                    | name                   |
+---------------------------------------|------------------------|
|                                      |                        |
+---------------------------------------+------------------------|
|                                      |                        |
| b779f01e-95d6-4ca6-a9ec-a37753f66a2b | sco-allinone-kvm       |
+---------------------------------------|------------------------|
|                                      |                        |
| SCO Core + additional KVM compute node deployment (Existing machines) | ACTIVE |
```
For more details about the required resources for the template, run the following command, for example:

```
source /root/keystonerc; ds template-resources-list b779f01e-95d6-4ca6-a9ec-a37753f66a2b
```

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>run_list</th>
<th>run_order</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>Existing Machine</td>
<td>role[sco-allinone-kvm-mgm]</td>
<td>1</td>
</tr>
<tr>
<td>compute</td>
<td>Existing Machine</td>
<td>role[kvm-compute]</td>
<td>2</td>
</tr>
<tr>
<td>iwd</td>
<td>Existing Machine</td>
<td>role[iwd]</td>
<td>3</td>
</tr>
</tbody>
</table>

## Customizing deployment parameters

The Deployment Service provides default values for most of the parameters needed to deploy an IBM Cloud Orchestrator. If the default values are not able to meet your environment, you can use the CLI or wizard to override these values.

To list the deployment template parameters, run the following command:

```
ds template-parameters-list <template_uuid>
```

where `<template_uuid>` is the ID of the template that you can get by running the `ds template-list` command.

For a list of the deployment parameters, see [“Deployment parameters” on page 43](#).

**Note:** The `SingleSignOnDomain` parameter is empty by default. You must set the `SingleSignOnDomain` parameter value to the DNS domain name where the manage-from components are installed. The manage-from components include IBM HTTP Server, IBM Cloud Orchestrator user interfaces, Business Process Manager, and Workload Deployer.

The DNS domain name is in the form `y.z` or `x.y.z`, as shown in the following examples:

```
example.com
host.example.com
```

Replace the example value with the name of your DNS domain.

You can specify the `SingleSignOnDomain` parameter value when you create the deployment job.

If you plan to use Neutron networks, consider that:

- The `MGMNetInterface` parameter represents an interface that must connect to a public and routable network.
- The `DATANetInterface` parameter represents an interface that must connect to a network dedicated for VXLAN data.
- The `EXTNetInterface` parameter represents an interface that is used to connect to a network for external access.
You can have the following three Neutron network topologies:

For this topology, MGMNetInterface = eth0, DATANetInterface = eth0 and EXTNetInterface = eth0.

For this topology, MGMNetInterface = eth0, DATANetInterface = eth1 and EXTNetInterface = eth0.
For this topology, MGMNetInterface = eth0, DATANetInterface = eth1, and EXTNetInterface = eth2.

Neutron network topology 1 is used for simple deployment, Neutron network topology 2 is used for production environment, and Neutron network topology 3 is used only in a complex environment.

**Note:**
- The NIC name must be the same for all nodes if they are all in the same network.
- The Compute Nodes can be either KVM hypervisor or ESXi servers.

### Deployment parameters

Check the list of all deployment parameters that you can configure before deploying IBM Cloud Orchestrator, and the parameter default values.

**Table 8. Deployment parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPMHome</td>
<td>/opt/ibm/BPM/v8.5</td>
<td>The installation path of Business Process Manager.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The path can be created or accessed by the user that deploys IBM Cloud Orchestrator.</td>
</tr>
<tr>
<td>CinderStatePath</td>
<td>/var/lib/cinder</td>
<td>The path to save the cinder state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Make sure that the cinder user can access the path.</td>
</tr>
<tr>
<td>CinderVGSize</td>
<td>100</td>
<td>All the volume size.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Make sure that the Region Server has enough disk space.</td>
</tr>
<tr>
<td>CinderVolumesDir</td>
<td>/var/lib/cinder/volumes</td>
<td>The path to save the volume.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Make sure that the cinder user can access the path.</td>
</tr>
<tr>
<td>Name</td>
<td>Default Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DATANetInterface</td>
<td>eth0</td>
<td>The VM data network interface. It is used only for Neutron networks.</td>
</tr>
<tr>
<td>DB2DataDir</td>
<td>/home/db2inst1</td>
<td>The folder to save DB2 data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The path can be created or accessed by the user that deploys IBM Cloud Orchestrator.</td>
</tr>
<tr>
<td>DB2DownloadDir</td>
<td>/tmp</td>
<td>The folder to save download binary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Change it if you do not want to put the file in /tmp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The path must exist and must be able to be written by the user that deploys IBM Cloud Orchestrator.</td>
</tr>
<tr>
<td>DB2ServiceIPNetmask</td>
<td>255.255.255.0</td>
<td>The Netmask of DB2 service IP address. It is used when you want to deploy IBM Cloud Orchestrator with an internal DB2, and give the DB2 node a secondary IP address for the future to move the DB2 to another machine.</td>
</tr>
<tr>
<td>DiscoveredVMPrefix</td>
<td>Onboarded</td>
<td>The prefix that will be added to the name of discovered instances. This parameter is only for a VMware Region.</td>
</tr>
<tr>
<td>DiscoveredFlavorPrefix</td>
<td>Onboarded</td>
<td>The prefix that will be added to the instance name for flavors created for discovered instances. This parameter is only for a VMware Region.</td>
</tr>
<tr>
<td>VMWsdlLocation</td>
<td></td>
<td>VMware wsdl location. Only need provide this location when you connect to vCenter 5.0. This parameter is only for a VMware Region.</td>
</tr>
<tr>
<td>EXTNetInterface</td>
<td>eth0</td>
<td>The external network interface. It is used only for Neutron networks.</td>
</tr>
<tr>
<td>GlanceFileStore</td>
<td>/var/lib/glance/images</td>
<td>The path to save the glance image. Make sure that the glance user can access the path.</td>
</tr>
<tr>
<td>IHSLocalDir</td>
<td>/tmp/ihsinstall</td>
<td>The temporary path for IBM HTTP Server to save local data. Change it if you do not want to put the file in /tmp. The path must exist and must be able to be written by the user that deploys IBM Cloud Orchestrator.</td>
</tr>
</tbody>
</table>
Table 8. Deployment parameters (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MGMNetInterface</td>
<td>eth0</td>
<td>The interface that is used for communication between the IBM Cloud Orchestrator management components. It must be consistent on each node.</td>
</tr>
<tr>
<td>NetworkManager</td>
<td>nova.network.manager.VlanManager</td>
<td>It is used when you want to deploy a nova-network based environment or region. Keep the value as it is. In special cases, this may also have the value of nova.network.manager.FlatManager.</td>
</tr>
<tr>
<td>NovaStatePath</td>
<td>/var/lib/nova</td>
<td>The path to save the nova instances. Make sure that the nova user can access the path.</td>
</tr>
</tbody>
</table>
| OrchestratorPassword  | passw0rd                                           | The password for all of the IBM Cloud Orchestrator components. The password can contain only the following characters:
<p>|                       |                                                    | Important: For security reasons, you should change the default value.                                                                      |
| OSLibvirtType         | kvm                                                | The virtualization type for KVM-based IBM Cloud Orchestrator deployment. Change it to qemu if you are using a virtual machine to deploy. Not recommended for production. |
| OSNetworkType         | nova (for non-neutron based template)              | The network type. No need to change it since it is associated with the template.                                                          |
|                       | neutron (for neutron-based template)               |                                                                                                                                              |
| PCGDownloadDir        | /tmp/pcg                                           | The temporary path for Public Cloud Gateway to save data. Change it if you do not want to put the file in /tmp. The path must exist and must be able to be written by the user that deploys IBM Cloud Orchestrator. |
| PCGHomeDir            | /opt/ibm/pcg                                       | The path for Public Cloud Gateway to save data. The path can be created or accessed by the user that deploys IBM Cloud Orchestrator.            |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegionName</td>
<td>RegionOne (all-in-one) RegionCentral (multiple region) RegionKVM (KVM region) RegionVMware (VMware region) RegionZVM (z/VM region) RegionPower (Power region)</td>
<td>This is the region name for OpenStack. It is important only when you want to deploy a multiple region environment. No need to change it for all-in-one based template. In a multiple region environment, region name must be unique.</td>
</tr>
<tr>
<td>SingleSignOnDomain</td>
<td></td>
<td>You must set the SingleSignOnDomain parameter value to the DNS domain name where the manage-from components are installed. The manage-from components include IBM® HTTP Server, IBM Cloud Orchestrator user interfaces, Business Process Manager, and Workload Deployer.</td>
</tr>
<tr>
<td>TSAAMLocal</td>
<td>/tsa</td>
<td>The path for TSA to put data. The path can be created or accessed by the user that deploys IBM Cloud Orchestrator.</td>
</tr>
<tr>
<td>VMBridge</td>
<td>br4896</td>
<td>The bridge name created in the hypervisor, only for nova-network.</td>
</tr>
<tr>
<td>VMClusterName</td>
<td></td>
<td>The cluster name in vCenter that is used for the virtual machine. This parameter is only for a VMware Region.</td>
</tr>
<tr>
<td>VMDataStoreName</td>
<td></td>
<td>The datastore cluster name in vCenter that is used to put the virtual machine. You can specify a regular expression to match the name of a datastore. For example, VMDataStoreName=&quot;nas.*&quot; selects all the datastores that have a name starting with &quot;nas&quot;.</td>
</tr>
<tr>
<td>VMDns1</td>
<td></td>
<td>The primary DNS used by nova-network. Change it to your corporate DNS. Must be specified as an IP address but not as the host name, for example: 10.178.144.228.</td>
</tr>
<tr>
<td>VMDns2</td>
<td></td>
<td>The secondary DNS used by nova-network. Change it to your corporate DNS. Must be specified as an IP address but not as the host name, for example: 10.178.144.228.</td>
</tr>
</tbody>
</table>
Table 8. Deployment parameters (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMEndIP</td>
<td>10.10.0.200</td>
<td>The final IP address of the virtual machine IP range that is used by IBM Cloud Orchestrator. It is only used by nova network.</td>
</tr>
<tr>
<td>VMGATEWAY</td>
<td>10.10.0.1</td>
<td>The gateway of the virtual machine network.</td>
</tr>
<tr>
<td>VMNETLABEL</td>
<td>demo</td>
<td>The label of the virtual machine network.</td>
</tr>
<tr>
<td>VMInterface</td>
<td>vmnic0</td>
<td>The VMware physical Ethernet adapter name for vlan networking. Change it to the actual adapter used. The nova network manager will create vlan port groups attached to this nic when required. This parameter is only for a VMware Region.</td>
</tr>
<tr>
<td>VMIPV4ADDR</td>
<td>10.10.0.0/24</td>
<td>The default subnet for nova-network. Change it according to your datacenter environment.</td>
</tr>
<tr>
<td>VMStartIP</td>
<td>10.10.0.100</td>
<td>The initial IP address of the virtual machine IP range that is used by IBM Cloud Orchestrator. It is only used by nova network.</td>
</tr>
<tr>
<td>VMServerHost</td>
<td></td>
<td>The VMware vCenter host. This parameter is only for a VMware Region.</td>
</tr>
<tr>
<td>VMServerUserName</td>
<td></td>
<td>The user name that is used to connect to VMware vCenter. This parameter is only for a VMware Region.</td>
</tr>
<tr>
<td>VMServerPassword</td>
<td></td>
<td>The password of the user specified in VMServerUserName.</td>
</tr>
<tr>
<td>VMDataStoreName</td>
<td></td>
<td>The datastore in vCenter that used to store the virtual machine. It support to specify a regex to match the name of a datastore. For example, VMDataStoreName=&quot;nas.*&quot; selects all the data stores that have a name starting with nas. This parameter is only for a VMware Region.</td>
</tr>
<tr>
<td>VMVLANID</td>
<td>100</td>
<td>Vlan ID of the virtual machine network.</td>
</tr>
<tr>
<td>VXLANMcastGroup</td>
<td>224.0.0.100</td>
<td>VXLAN multicast group. It is used only for Neutron networks.</td>
</tr>
<tr>
<td>VXLANVniRange</td>
<td>1000:1600000</td>
<td>VXLAN VNI range. It is used only for Neutron networks.</td>
</tr>
<tr>
<td>WorkloadDeployerTmpPath</td>
<td>/tmp/iwd</td>
<td>The temporary path for Workload Deployer to save binary. The path can be created or accessed by the user that deploys IBM Cloud Orchestrator.</td>
</tr>
</tbody>
</table>
Quota related installation parameters

It is possible to alter the default quota values for OpenStack tenants at installation time. When deploying IBM Cloud Orchestrator to VMware environments with existing workload, it is possible to exceed the default quotas based upon discovered workload. In that situation, or other situation where the default quotas are known to be insufficient at installation time, the following Region Server related parameters can be used:

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuotaCores</td>
<td>20</td>
<td>Quota for cores</td>
</tr>
<tr>
<td>QuotaDriver</td>
<td>nova.quota.DbQuotaDriver</td>
<td>Quota for driver</td>
</tr>
<tr>
<td>QuotaFixed_ips</td>
<td>-1</td>
<td>Quota for fixed IP addresses</td>
</tr>
<tr>
<td>QuotaFloating_ips</td>
<td>10</td>
<td>Quota for floating IP addresses</td>
</tr>
<tr>
<td>QuotaInjectedFileContentBytes</td>
<td>10240</td>
<td>Quota for injected file content bytes</td>
</tr>
<tr>
<td>QuotaInjectedFilePathBytes</td>
<td>255</td>
<td>Quota for injected file path bytes</td>
</tr>
<tr>
<td>QuotaInjectedFiles</td>
<td>5</td>
<td>Quota for injected files</td>
</tr>
<tr>
<td>QuotaInstances</td>
<td>10</td>
<td>Quota for instances</td>
</tr>
<tr>
<td>QuotaKeyPairs</td>
<td>100</td>
<td>Quota for key pairs</td>
</tr>
<tr>
<td>QuotaMetadataItems</td>
<td>128</td>
<td>Quota for metadata items</td>
</tr>
<tr>
<td>QuotaRam</td>
<td>51200</td>
<td>Quota for RAM</td>
</tr>
<tr>
<td>QuotaSecurityGroupRules</td>
<td>20</td>
<td>Quota for security group rules</td>
</tr>
<tr>
<td>QuotaSecurityGroups</td>
<td>50</td>
<td>Quota for security group</td>
</tr>
</tbody>
</table>

Configuring an external database

You can configure an external database server to be used by IBM Cloud Orchestrator.

After the Deployment Service has been installed, you can deploy IBM Cloud Orchestrator to make use of your own external database. IBM Cloud Orchestrator does not create its own database server instance but leverages the external database server. The external database must be DB2 version 10.5.0.2 or higher.

**Note:** For the high availability topology, an external database is a mandatory prerequisite.

To configure an external database, perform the following steps:
1. "Prepare the databases for IBM Cloud Orchestrator."
2. "Create database nodes for IBM Cloud Orchestrator" on page 50.
3. "Run the deployment job" on page 52.

Prepare the databases for IBM Cloud Orchestrator

Perform the following steps:
1. From the installer package, copy the data/orchestrator-chef-repo/packages/workload_deployer/archive.zip and installer/create_dbs.sh files to a directory of your choice on the database machine.
2. Log in to the database machine as root and give executable permission to the script `create_dbs.sh`.

3. Set the following environment variables before executing `create_dbs.sh`:
   
   ```
   export DB2_INSTANCE_USER=<your db2 instance user name, for example db2inst1>
   export DB2_INSTANCE_PWD=<your db2 instance password>
   export USER_DEFAULT_PWD=<default password of database users>
   ```

4. Run `./create_dbs.sh all`

   This creates databases for the Central Server and a single Region Server with default values.

   If you just want to create databases for the Central Server on this machine, run:
   
   `./create_dbs.sh central`

   If you just want to create databases for a single Region Server on this machine, run:
   
   `./create_dbs.sh region`

   If you just want to create databases for multiple Region Servers on this machine assumed you want to deploy three regions, run:
   
   `./create_dbs.sh region=1,2,3`

   If you want to add a database for a new Region Server on this machine, for example, a fourth Region Server, run:
   
   `./create_dbs.sh region=4`

   The script will use the default environment variable to create the database, user and schema against the specific DB2 instance on the database server. The default values of the configuration are defined in the script and can be override by export system environment variables before running the `create_dbs.sh`. If you create a database for multiple regions, it will use the same database with different user/schema in the database server: for example, the `create_dbs.sh region` will create `nova` user in system and database, `create_dbs.sh region=1,2,3` will create `nova1`, `nova2`, `nova3` in the system and database for each region.

   Be aware that the script assumes that the database and user/schema should not exist before you run the script, you must check the DB2 server to make sure that there is not conflict.

The following are the default environment variables that are used in `create_dbs.sh`:

- `OS_DB_NAME=<OpenStack DB Name, default=OPENSTAC>`
- `DB2_INSTANCE_PORT=<DB2 instance port, default=50000>`
- `OS_NOVA_DB_NAME=<OpenStack Nova DB User, default=OPENSTAC>`
- `OS_NOVA_DB_USER=<OpenStack Nova DB User, default=nova>`
- `OS_NOVA_DB_PWD=<OpenStack Nova DB Password>`
- `OS_CINDER_DB_NAME=<OpenStack Cinder DB Name, default=OPENSTAC>`
- `OS_CINDER_DB_USER=<OpenStack Cinder DB User, default=cinder>`
- `OS_CINDER_DB_PWD=<OpenStack Cinder DB Password>`
- `OS_HEAT_DB_NAME=<OpenStack Heat DB Name, default=OPENSTAC>`
- `OS_HEAT_DB_USER=<OpenStack Heat DB User, default=heat>`
- `OS_HEAT_DB_PWD=<OpenStack Heat DB Password>`
- `OS_DASH_DB_NAME=<OpenStack Dashboard DB Name, default=OPENSTAC>`
- `OS_DASH_DB_USER=<OpenStack Dashboard DB User, default=dash>`
- `OS_DASH_DB_PWD=<OpenStack Dashboard DB Password>`
- `OS_KEYSTONE_DB_NAME=<OpenStack Keystone DB Name, default=OPENSTAC>`
- `OS_KEYSTONE_DB_USER=<OpenStack Keystone DB User, default=keystone>`
- `OS_KEYSTONE_DB_PWD=<OpenStack Keystone DB Password>`
- `OS_NEUTRON_DB_NAME=<OpenStack Neutron DB Name, default=OPENSTAC>`
- `OS_NEUTRON_DB_USER=<OpenStack Neutron DB User, default=neutron>`
- `OS_NEUTRON_DB_PWD=<OpenStack Neutron DB Password>`
- `OS_CEIL_DB_NAME=<OpenStack Ceilometer DB Name, default=OPENSTAC>"
OS_CEIL_DB_USER=<OpenStack Ceilometer DB User, default=ceil>
OS_CEIL_DB_PWD=<OpenStack Ceilometer DB Password>
OS_CEIL_DB_NOSQLPORT=<Mongo DB Port, default=27017>
OS_GLANCE_DB_NAME=<OpenStack Glance DB Name, default=OPENSTAC>
OS_GLANCE_DB_USER=<OpenStack Glance DB User, default=glance>
OS_GLANCE_DB_PWD=<OpenStack Glance DB Password>
BPM_DB_NAME=<BPM DB Name, default=BPMDB>
BPM_DB_USER=<BPM DB User, default=bpmuser>
BPM_DB_PWD=<BPM DB Password>
CMN_DB_NAME=<CMN DB Name, default=CMNDB>
CMN_DB_USER=<CMN DB User, default=bpmuser>
CMN_DB_PWD=<CMN DB Password>
PDW_DB_NAME=<PDW DB Name, default=PDWDB>
PDW_DB_USER=<PDW DB User, default=bpmuser>
PDW_DB_PWD=<PDW DB Password>

Note:
- Database name length must be no more than eight characters, for example, OPENSTAC and not OPENSTACK.
- For Business Process Manager databases (BPMDB, CMNDB and PDWDB), if you use the same user name for different databases, make sure that you use the same password too. Do not use different passwords for the same user.
- Make sure to set the USER_DEFAULT_PWD or the default password is password.
- Make sure to set correctly the firewall to allow the DB2 communication. The default port numbers are 50000 and 27017.
- If you need to restart the DB, after restarting, you need to restart NoSQL by following the procedure described in "Restarting NoSQL" on page 187.

The following is an example of the output of create_dbs.sh:

```
{ Address: <HOSTNAME>, Port: 50000, Name: OPENSTAC, User: nova, Password: <PASSWORD>, Fqdn: <HOSTNAME>}
{ Address: <HOSTNAME>, Port: 50000, Name: OPENSTAC, User: dash, Password: <PASSWORD>, Fqdn: <HOSTNAME>}
```

If you modify it slightly, you can use it to create a database node in the deployment service, for example:

ds node-create -t IBM:SCO::Database
-p '{ Address: <HOSTNAME>, Port: 50000, Name: OPENSTAC, User: nova, Password: <PASSWORD>, Fqdn: <HOSTNAME>}'
nova_db

Create database nodes for IBM Cloud Orchestrator

Use the ds template-list command to identify the template that best fits your need. The names of the templates for external database ends with -extdb, for example, sco-allinone-extdb and sco-central-region-extdb. Perform the following steps:

1. Log on to the Deployment Server and source the following file: source keystonerc otherwise the ds... commands will all fail.
2. Identify the nodes to be registered by looking at the resource names of type Existing Database in the output of the ds template-resources-list command.

For example:

ds template-resources-list sco-allinone-extdb

```
+-----------------+-----------------+-------------------+-----------+
| name            | type            | run_list          | run_order |
+-----------------+-----------------+-------------------+-----------+
| control         | Existing Machine| role[sco-allinone-extdb] | 1         |
| compute         | Existing Machine| role[kvm-compute]  | 2         |
| bpm_db          | Existing Database|                   |           |
```
3. List the node type by running the following command:
   
   ds node-type-list
   
   +-------------------+
   | type              |
   +-------------------+
   | IBM::SCO::Node    |
   | IBM::SCO::Database|
   +-------------------+

4. Show the node properties by running the following command:
   
   ds node-type-show -p IBM::SCO::Database
   
   +------------+---------+--------+----------------+------------------------------------+
   | Properties | Default | Type   | Update | Description    |
   |------------|---------|--------|--------|----------------+------------------------------------|
   | Name       |         | string | True   | The database name |
   | NoSqlPort  | 27017   | number | True   | The database no SQL port |
   | Fqdn       |         | string | True   | The FQDN of the database machine |
   | User       |         | string | True   | The database schema user |
   | Address    |         | string | True   | The Address of the database machine |
   | Password   |         | string | True   | The database password |
   | Port       | 50000   | number | True   | The database port |
   +------------+---------+--------+----------------+------------------------------------+

Create database nodes for all the existing database resources by running the following commands (each command on one line):
   
   ds node-create -t IBM::SCO::Database
   -p '{Address: 192.168.51.147, User: cinder, Password: admin, Name: openstack}'
   
   ds node-create -t IBM::SCO::Database
   -p '{Address: 192.168.51.147, User: nova, Password: admin, Name: openstack}'
   
   ds node-create -t IBM::SCO::Database
   -p '{Address: 192.168.51.147, User: dash, Password: admin, Name: openstack}'
   
   ds node-create -t IBM::SCO::Database
   -p '{Address: 192.168.51.147, User: glance, Password: admin, Name: openstack}'
   
   ds node-create -t IBM::SCO::Database
   -p '{Address: 192.168.51.147, User: keystone, Password: admin, Name: openstack}'
   
   ds node-create -t IBM::SCO::Database
   -p '{Address: 192.168.51.147, User: ceil, Password: admin, Name: openstack}'
   
   ds node-create -t IBM::SCO::Database
   -p '{Address: 192.168.51.147, User: neutron, Password: admin, Name: openstack}'
   
   ds node-create -t IBM::SCO::Database
   -p '{Address: 192.168.51.147, User: heat, Password: admin, Name: openstack}'
ds node-create -t IBM::SCO::Database
    -p '{Address: 192.168.51.147, User: bpmuser, Password: password, Name: CMNDB}'
cmn_db

ds node-create -t IBM::SCO::Database
    -p '{Address: 192.168.51.147, User: bpmuser, Password: password, Name: BPMDB}'
bpm_db

ds node-create -t IBM::SCO::Database
    -p '{Address: 192.168.51.147, User: bpmuser, Password: password, Name: PDWDB}'
pdw_db

You can verify that the nodes were successfully created by running the ds node-list command.

**Run the deployment job**

The following section on how to create and run a deployment job is just an example. If you are performing the database setup for a high availability environment, follow the instructions on ["Installing the Central Servers" on page 71](#).

Before running a deployment job, create it by running the following command (on one line), for example:

ds job-create -N "dashboard_db=dashboard_db;compute_db=compute_db;cmn_db=cmn_db;bpm_db=bpm_db;
control=control;metering_db=metering_db;network_db=network_db;
orchestration_db=orchestration_db;compute=compute;image_db=image_db;pdw_db=pdw_db;
keystone_db=keystone_db"
    -t sco-allinone-extdb sco-allinone-extdb

Run the job by using the ds job-execute command.

**Deploying an IBM Cloud Orchestrator environment**

Deploy an IBM Cloud Orchestrator environment according to your business needs.

To deploy IBM Cloud Orchestrator, perform the following steps:

1. Register the resources, virtual or physical machines, in the Deployment Service.
2. Select the IBM Cloud Orchestrator to deploy.
3. Provide the deployment parameters.
4. Run the deployment job.
5. Extend the existing IBM Cloud Orchestrator environment.

You can run the previous procedure by using either a console-based wizard or the command line interface.

**Note:** After you completed the environment deployment, you must assign the installed availability zone to the Default domain and the admin project. For more information, see ["Assigning a zone to a domain" on page 219](#) and ["Assigning a zone to a project" on page 224](#).

Before starting the IBM Cloud Orchestrator deployment, it is important to understand the concepts and artifacts that are used by the Deployment Service:

**Node** The resource that is used to deploy or configure IBM Cloud Orchestrator. There are two types of nodes, one is a general node that is used to represent a virtual machine or physical machine, the other one represents the database, in case you want to use an external DB2.
Template
The deployment template that is used to define different topologies of IBM Cloud Orchestrator.

Parameter
The parameter inside the template. It is used to customize the deployment of IBM Cloud Orchestrator, for example to define the network type or the passwords.

Deployment job
used to define the deployment of the IBM Cloud Orchestrator environment. It can be a hierarchy if two jobs are related, for example, Central Server and Region Server jobs (the Region Server jobs depend on the Central Server job).

Note: Non-English characters are not supported in the Node, Template, Parameter and Deployment job name. Make sure to use English names for those resources.

Using the Deployment Service wizard to deploy IBM Cloud Orchestrator
The Deployment Service wizard wraps all the CLI commands with a console-based UI that you can use to deploy IBM Cloud Orchestrator.

You can select a deployment template and customize it. For information about customizing template parameters, see “Customizing deployment parameters” on page 41.

To start the wizard, run the following command on the Deployment Server:

- For a root user:
  source /root/openrc; ds wizard

- For a non-root user, create an openrc file in your home with content:
  ```
  export OS_USERNAME=admin
  export OS_PASSWORD=$(openstack-obfuscate -u <obfuscated password goes here>)
  export OS_TENANT_NAME=admin
  export OS_AUTH_URL=http://<fqdn or ip address of deployment service node>:5000/v2.0
  export OS_REGION_NAME=Deployment
  ```

  You can run all the ds command after source this file:
  ```
  source openrc; ds wizard
  ```

Note:
- The terminal window must be at least 35 lines and 130 columns.
- You can use also specify the -f parameter when running the ds wizard command, to generate a file to record the commands used by the wizard.

Choose the following option to deploy IBM Cloud Orchestrator:

0] New IBM Cloud Orchestrator deployment.
  - Start a new IBM Cloud Orchestrator deployment.

and follow the interactive procedure by entering the required information. For information about the deployment parameters, see “Deployment parameters” on page 43.

Note: When using the Deployment Service wizard, if you do not create correctly a job and you want to delete it, you must use the ds command. You cannot delete a
job by using the wizard.

**Using the command-line interface to deploy IBM Cloud Orchestrator**

You can use the command-line interface to deploy your IBM Cloud Orchestrator environment.

Before registering the nodes, you must prepare a virtual machine or physical machine that has Red Hat Enterprise Linux installed and SSH enabled. After the nodes are prepared, you can use the `ds node-create` command to register the nodes into the Deployment Service.

The deployment is done in two steps:

1. **“Registering the nodes.”** In the Deployment Service, define the specifications of the nodes to build the defined topology, such as the node IP address, the user to be used to connect to the node and the related password, the port to be used by the SSH protocol, the name of the network interface, and all the parameters that do not have a default value or for which you want to change the default value.

2. **“Running the deployment” on page 56**

After the initial deployment, you can add Region Servers to your environment as described in **“Deploying a Region Server” on page 57**.

**Registering the nodes**

To create a node, run the `ds node-create` command on the Deployment Server. You can specify the following parameters:

- **Address**
  The IP address or FQDN of the node. This parameter is optional if the `Fqdn` parameter is specified. If the IP address is not specified, the FQDN is used to connect to the node. If the `Fqdn` parameter is specified and the `Address` parameter is specified with an IP address, the `<Fqdn, IP address>` entry is added into the `/etc/hosts` file for the DNS service. If you are using Corporate DNS, make sure that the IP address of the node is able to be resolved by DNS and it is the same as the FQDN that you specified. To check if the node is registered in the Corporate DNS, you can use the `host <ip-addr>` address to check that the correct FQDN of the node is returned.

- **Fqdn**
  The FQDN of the node. This parameter is optional if the `Address` parameter is specified. If the FQDN is not specified, the installer tries to retrieve it automatically according to the specified address.

- **Port**
  The SSH port to access the node. This parameter is required.

- **ServiceAddress**
  The service IP address for DB2. This parameter is optional.

- **User**
  The user name to access the node via SSH. The user must be the root user or must have sudo privilege. This parameter is required.

- **Password**
  The password used to access the node. This parameter is optional if the `KeyFile` parameter is specified. The password can contain only the following characters:
  a-zA-Z0-9_
KeyFile

The key file location used to access the node. This parameter is optional if the Password parameter is specified.

To understand the number of nodes that are required in the selected topology, can use `ds template-resource-list <template_UUID>` command.

For example, if you deploy an environment by using the `sco-allinone-kvm` template, you must run the following commands on the Deployment Server:

```bash
source keystonerc; ds template-list
```

```
<table>
<thead>
<tr>
<th>b779f01e-95d6-4ca6-a9ec-a37753f66a2b</th>
<th>sco-allinone-kvm</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCO Core + additional KVM compute node deployment (Existing machines)</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>2014-03-27T01:30:54.661581</td>
<td>2014-03-27T06:58:35.211096</td>
</tr>
</tbody>
</table>
```

```bash
ds template-resources-list b779f01e-95d6-4ca6-a9ec-a37753f66a2b
```

+-------------------+-------------------+-----------------+-----------+
| name | type | run_list | run_order |
+-------------------+-------------------+-----------------+-----------+
| control | Existing Machine | role[sco-allinone-kvm-mgm] | 1 |
| compute | Existing Machine | role[kvm-compute] | 2 |
| iwd | Existing Machine | role[iwd] | 3 |
+-------------------+-------------------+-----------------+-----------+

```bash
ds node-create -t "IBM::SCO::Node" -p "{Address: 172.16.132.211, Port: 22, User: root, Password: password, Fqdn: sco-core.example.com}" sco-core
ds node-create -t "IBM::SCO::Node" -p "{Address: 172.16.132.212, Port: 22, User: root, Password: password, Fqdn: sco-iwd.example.com}" sco-iwd
ds node-create -t "IBM::SCO::Node" -p "{Address: 172.16.132.213, Port: 22, User: root, Password: password,Fqdn: sco-region.example.com}" kvm-compute
```

where `sco-core`, `sco-iwd`, and `kvm-compute` are the node names that you assign to each node. It is recommended to assign names that are related to the role of the node.

**Note:** When deploying a DB2 node, specify a service IP address. This is a mandatory requirement if you want to move your database on a different system at a later time. For information about moving the database, see "Move the database to an external system" on page 102. For example, run the following command (on one line):

```bash
ds node-create -t "IBM::SCO::Node" -p "{Address: 172.16.2.60, ServiceAddress: 172.16.2.245, User: root, Password: passw0rd}" db2-node
```

where 172.16.2.60 is the real IP address of the Central Server 1 node (where DB2 is deployed), and 172.16.2.245 is the service IP address of DB2. During the installation procedure, the service IP address is configured on the interface specified with the `MGMNetInterface` parameter, and the database connection of the other services is configured to use this service IP address.

Some templates may include **Resource Reference** type resource which refers to an existing Resource. For example:

```bash
ds template-resources-list fc11c554-24d4-4bba-a891-990aee1f310
```

+-------------------+-------------------+-----------------+-----------+
| name | type | run_list | run_order |
+-------------------+-------------------+-----------------+-----------+
| central_server_1 | Existing Machine | role[central-server-1] | 1 |
| central_server_2 | Existing Machine | role[central-server-2] | 2 |
+-------------------+-------------------+-----------------+-----------+
It is not needed to associate the node with this kind of resource when creating job. Otherwise, an error will be reported.

You can verify that the node registration completed successfully by using the following command:

ds node-list

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>address</th>
<th>user</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>9992bd06-0ae2-460b-aed0-0259a8acca70</td>
<td>sco-core</td>
<td>172.16.132.211</td>
<td>root</td>
<td>FREE</td>
</tr>
<tr>
<td>e5b6d881-1a66-4ac0-824f-d48c9296eae23</td>
<td>kvm-compute</td>
<td>172.16.132.213</td>
<td>root</td>
<td>FREE</td>
</tr>
<tr>
<td>f80fd22c-e7a5-40c2-be17-9c7bd60a0142</td>
<td>sco-iwd</td>
<td>172.16.132.212</td>
<td>root</td>
<td>FREE</td>
</tr>
</tbody>
</table>

Running the deployment

Run the following steps:

1. In the Deployment Service, register the deployment job by running the `ds job-create` command. For example:

   `ds job-create -t #{template_id}
   -P "MGMNetInterface=#{net_interface};SingleSignOnDomain=#{domain_name}"
   -N "control=#{sco-core-id};iwd=#{sco-iwd-id};compute=#{kvm-compute-id}"
   myallinonekvm`

where

- `template_id` is the ID of the `sco-allinone-kvm` template that you can get by using the `ds template` command.
- `net_interface` is the NIC device that must be the same in each node.
- `SingleSignOnDomain` is the DNS domain name where the manage-from components are installed.
- `sco-core-id`, `sco-iwd-id`, and `kvm-compute-id` are the IDs that you can get by using the `ds node-list` command.
- `myallinonekvm` is the name you choose to identify the job.

**Note:** In the Deployment Service commands, use double quotation marks to specify parameter values. Passwords can contain only the following characters: a-zA-Z0-9_

Spaces are not allowed. Use leading and trailing single quotation marks ('`) when special characters are used within a password, as shown in the following example:

`ds node-create -t "IBM::SCO::Node" -p "{Address: 172.16.132.211, Port: 22, User: root, Password: 'pass_w0rd', Fqdn: sco-core.example.com}" sco-core`
Note: When creating central servers and you want to use your own defined password, specify the OrchestratorPassword parameter as in the following example:

```
    ds job-create -t 326ade67-5991-46b6-b125-4c0721cb507c
                -N "central_server_1=b798236-4d46-4a7b-8816-4beb87647577;
                     central_server_2=ce1e0cde-d4f9-4ed6-9b71-744513997f5f;
                     central_server_3=466b29ab-9442-4a95-b2c0-00af17c6078b"
                -P "OrchestratorPassword=myownpassw0rd"
```

If a problem occurs during job creation and the job status is WARNING or ERROR, run the `ds job-show <job ID>` command to inspect the details of the job which also include the related error message.

2. To perform the installation, execute the deployment job by running the following command, for example:

```
    ds job-execute myallinonekvm
```

   where `myallinonekvm` is the name to identify the job that you specified in the previous step.

You can use the `ds job-list` command to check the status of a job. A FINISHED status indicates that the deployment finished.

**Deploying a Region Server**

After the initial deployment, you can add a Region Server to your environment.

You can add any of the following Region Servers:

- KVM region with nova network
- KVM region with neutron network
- VMware region with nova network
- VMware region with neutron network
- PowerVC region with neutron network
- z/VM region with neutron network

The deployment procedure is the same as described in “Using the command-line interface to deploy IBM Cloud Orchestrator” on page 54 with the difference being that, when deploying the Region Server, you must specify the job ID of the Central Server.

You can also deploy a Region Server by using the Deployment Service wizard as described in “Using the Deployment Service wizard to deploy IBM Cloud Orchestrator” on page 53 and choosing the following option:

   - Modify an IBM Cloud Orchestrator deployment, add region server or KVM compute node.

Follow the interactive procedure by entering the required information. For information about the deployment parameters, see “Deployment parameters” on page 43.

Note: When using the Deployment Service wizard, if you do not create correctly a job and you want to delete it, you must use the `ds` command. You cannot delete a job by using the wizard.
Installing a KVM Region Server

 Install a KVM Region Server in IBM Cloud Orchestrator.

The deployment procedure is the same as described in “Using the command-line interface to deploy IBM Cloud Orchestrator” on page 54 with the difference being that, when deploying the Region Server, you must specify the job ID of the Central Server. For example, to add a KVM region with a neutron network to an existing Central Server, run the following commands:

```
ds node-create -t "IBM::SCO::Node" \
- p "(Address: 172.16.132.219, Port: 22, User: root, Password: password )" kvm-region

nds node-create -t "IBM::SCO::Node" \

nds node-create -t "IBM::SCO::Node" \
- p "(Address: 172.16.132.221, Port: 22, User: root, Password: password )" kvm-compute

nds job-create -t #{template_id} -P "MGMNetInterface=#{net_interface}" \
- N "kvm_region_neutron=#{kvm-region-id};neutron_network_node=#{kvm-network-node}; \
kvm_compute=#{kvm-compute-id}" \
- p <central-server-job-id> kvm-region-server-neutron
```

When you create a job to deploy the Region Server, you must specify the Central Server job ID as the parent of Region Server job with the following option:

```
- p <central-server-job-id>
```

You can get the job ID by running the `ds job-list` command.

If a problem occurs during job creation and the job status is WARNING or ERROR, run the `ds job-show <job ID>` command to inspect the details of the job which also include the related error message.

**Note:**

If you are using the template of a Region Server with shared database mode (the Region Server uses the database deployed in the Central Server), you must include additional parameters in the `ds job-create` command, to specify different database user names:

```
nds job-create -t <template_id> \
- P "MGMNetInterface=<net_interface>; \
ComputeDBUsername=<compute_database_username>; \
DashboardDBUsername=<dashboard_database_username>; \
ImageDBUsername=<image_database_username>; \
MeteringDBUsername=<metering_database_username>; \
NetworkDBUsername=<network_database_username>; \
OrchestrationDBUsername=<orchestration_database_username>; \
VolumeDBUsername=<volume_database_username>" \
- N "kvm_region_neutron=#{kvm-region-id};neutron_network_node=#{kvm-network-node}; \
kvm_compute=#{kvm-compute-id}" \
- p <central-server-job-id> \
kvm-region-server-neutron
```

**Example:**

```
ds job-create -t fbf9b885-2f72-4da6-930b-44f792925e29 \
- P "MGMNetInterface=eth0;ComputeDBUsername=nova2; \
DashboardDBUsername=dash2;ImageDBUsername=glance2; \
MeteringDBUsername=ceil2;NetworkDBUsername=neutron2; \
OrchestrationDBUsername=heat2;VolumeDBUsername=cinder2" \
- N "kvm_region_neuron=e76db559-2605-4e68-b69b-4de89ccdedb5; \
```
Installing a VMware Region Server
This topic describes how to install a VMware Region Server.

Before you begin
Be sure that the Deployment Server and the Central Servers are ready in your environment. For more information, see "Installing the Deployment Service" on page 37 and "Deploying an IBM Cloud Orchestrator environment" on page 52.

Procedure
1. Choose the template. Run the following command to get the templates related to VMware. In this procedure the template vmware_region_neutron is used as an example.
   ```bash
ds template-list | grep vmware
   ...
   | e9467b25-fc7f-4d0b-ab4d-fb052768a9fc | vmware_region_neutron
   | VMware region-server deployment with Neutron network.
   ...
   ```

2. Check the template ID and the related resources and parameters by running the following command, for example:
   ```bash
ds template-resources-list e9467b25-fc7f-4d0b-ab4d-fb052768a9fc
   +----------------------+--------------------+------------------------------------------+-----------+
   | name                 | type               | run_list                                 | run_order |
   +----------------------+--------------------+------------------------------------------+-----------+
   | vmware_region_server | Existing Machine   | role[vmware-region],role[vmware-compute] | 1         |
   | neutron_network_node | Existing Machine   | role[neutron-network-node]               | 2         |
   | vrs_ico_agents_sar   | Resource Reference | role[ico_mon_agent_sar]                  | 3         |
   | nnn_ico_agents_sar   | Resource Reference | role[ico_mon_agent_sar]                  | 4         |
   +----------------------+--------------------+------------------------------------------+-----------+

3. Create nodes. According to the resources of the vmware_region_neutron template, you must create the vmware_region_server and the neutron_network_node nodes. Run the following commands, for example:
   ```bash
ds node-create -t 'IBM::SCO::Node' \
   -p '{Address: 172.16.132.254, Port: 22, User: root, Password: password }' \
   vmwareregion

ds node-create -t 'IBM::SCO::Node' \
   -p '{Address: 172.16.132.253, Port: 22, User: root, Password: password }' \
   vmwareneutron
   ```

4. Check the node IDs by running the following command:
   ```bash
ds node-list
   +--------------------------------------+----------------+--------+
   | id | name         | type   | status |
   +--------------------------------------+----------------+--------+
   | 0be0d025-8637-4ec9-a51d-a8b1065aae72 | vmwareregion  | INUSE  |
   | bfb2a2cc-dc4f-4c43-aebf-be07567f3e19 | vmwareneutron | INUSE  |
   | 5d9f6dd8-683f-43ce-99d1-183636414670 | central1     | INUSE  |
   | 699261dc-b57b-453b-9c92-00b2d202a750 | central1     | INUSE  |
   | 7dbd222f-d20a-4pdb-8f3e-77fd66067520 | central1     | INUSE  |
   +--------------------------------------+----------------+--------+
   +--------------------------------------+----------------+--------+
   | job_id | resource   | created_at |
   +--------------------------------------+----------------+--------+
   | 11dae147-86a0-49be-8743-46146e687bde | vmware_region_server | 2014-08-13T03:41:41.400463 |
   | 11dae147-86a0-49be-8743-46146e687bde | neutron_network_node | 2014-08-13T03:41.993816 |
   | 103cb7-30-54-c7-b0b2-8de9d051ad1d | central_server_3 | 2014-08-13T03:41.38.865586 |
5. Make sure that the prerequisite job to install the Central Servers completed successfully by running the following command, for example:

```bash
ds job-list
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>status</th>
<th>pjob_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>103cbe78-3015-4c7c-b02b-8de9d051ad1d</td>
<td>centralsrvjob</td>
<td>FINISHED</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>created_at</th>
<th>updated_at</th>
</tr>
</thead>
</table>

6. Create a job to install the VMWare region server and the neutron server by running the following command, for example:

```bash
ds job-create -t #{template_id} \
-P "VMNetInterface=#{net_interface};VMServerHost=#{def_vcenter_host};" \
VMServerPassword=#{def_password};VMServerUsername=#{def_user};" \
VMDataStoreName=#{def_data_store};VMClusterName=#{def_cluster_name}" \
-N "vmware_region_server=#{vmware-region-id};" \
neutron_network_node=#{vmware-network-node-id}" \
-p <central-server-job-id> \
vmware-region-server-neutron
```

where

- **template_id**
  Can be get from ds template. Select the template with name vmware_region_neutron.

- **net_interface**
  Is the NIC device. In each node it must be the same.

- **VMServerHost**
  Is the vCenter host IP address.

- **VMServerUsername**
  Is the user name that is used to connect to the vCenter.

- **VMServerPassword**
  Is the password for VMServerUsername.

- **VMDataStoreName**
  Is the data store that is used to store the virtual machine. You can specify a regular expression to match the name of a datastore. For example, VMDataStoreName="nas.*" selects all the data stores that have a name starting with nas.

- **VMClusterName**
  Is the cluster name in vCenter that use to put the virtual machine.

- **vmware-region-id**
  Can be get from ds node-list with the name you specified.

- **vmware-network-node-id**
  Can be get from ds node-list with the name you specified.
central-server-job-id
  Can be get from ds job-list.

**Note:** If you are using the template of a Region Server with shared database mode (the Region Server uses the database deployed in the Central Server), you must include the following additional parameters in the-P option of the ds job-create command:

ComputeDBUsername=<compute_database_username>
DashboardDBUsername=<dashboard_database_username>
ImageDBUsername=<image_database_username>
MeteringDBUsername=<metering_database_username>
NetworkDBUsername=<network_database_username>
OrchestrationDBUsername=<orchestration_database_username>
VolumeDBUsername=<volume_database_username>

The following output is displayed, for example:

```
+--------------------------------------+------------------------------+----------
| id | name | status            |
| id |     | status            |
| 11dae147-86a0-498e-87d3-46146e687bde | vmware-region-server-neutron | CREATED |
```

If a problem occurs during job creation and the job status is WARNING or ERROR, run the ds job-show <job ID> command to inspect the details of the job which also include the related error message.

7. Run the job to install the VMware Region Server by running the following command, for example:

```
ds job-execute 11dae147-86a0-498e-87d3-46146e687bde
```

**Installing a PowerVC Region Server**
Install a PowerVC Region Server in IBM Cloud Orchestrator.

**Before you begin**
Ensure that the Deployment Service server and the Central Servers are ready in IBM Cloud Orchestrator.

**Note:** Using a PowerVC server that has been configured for both Shared Storage Pools and SAN-fabric managed Storage is not recommended. If possible, you should use two PowerVC servers, each configured for using a single Storage Type, and two PowerVC Region Servers to manage them.

**Procedure**
1. Create nodes according to the resources of the powervc_region_neutron template. You must create the powervc_region_server and neutron_network_node nodes.

**powervc_region_server node:**

```
ds node-create -t 'IBM::SCO::Node' -p '{Address: 192.17.69.15, Port: 22, User: root, Password: passw0rd, Fqdn: PVCRS.69.customer.ibm.com}' powervc_region_server
```

**neutron_network_node node**

```
ds node-create -t 'IBM::SCO::Node' -p '{Address: 192.17.69.16, Port: 22, User: root, Password: passw0rd, Fqdn: PVCNS.69.customer.ibm.com}' neutron_network_node
```
2. Check the node IDs by running the following command:

```
ds node-list
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>type</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>6b391675-2eb5-4757-80ca-a01c8463f7</td>
<td>central_server_2</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
<tr>
<td>8b598b0-140a-4cb5-ade7-b6d7b7b3f9b</td>
<td>central_server_3</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
<tr>
<td>d61cc70-0e0e-4f1d-a3dd-b63297153f8b</td>
<td>power_region_server</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
<tr>
<td>d95ffae-d7ec-4826-ac99-65a83b0a06</td>
<td>neutron_network_node</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
<tr>
<td>e18c685a-529c-4f4c-9e4b-6da049f87f27</td>
<td>central_server_1</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
</tbody>
</table>

```

3. Check the template ID, related resources and parameters by running the following commands:

- `ds template-list`
- `ds template-resources-list <template id>`
- `ds template-parameters-list <template id>`

4. Ensure the prerequisite job to install the Central Servers already exists by running the following command:

```
ds job-list
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>6105da45-5af7-4aec-b823-efa9647af28</td>
<td>central_servers_install</td>
<td>FINISHED</td>
</tr>
</tbody>
</table>

```

5. Create a job to install the PowerVC region server by using the `ds job-create` command. The following parameters are mandatory:

- **PVCServer**
  The location of the PowerVC server
  Default: null

- **PVCServerUser**
  The root user of the PowerVC server
  Default: null

- **PVCServerPassword**
  The root password for the PowerVC server
  Default: null

- **PVCQpidPassword**
  The qpid password for the PowerVC server
This password is found by logging into PowerVC server and performing the following command:

```
cat /opt/ibm/powervc/data/pw.file
```

Default: null

**Note:** Identify the parent job that installs the central servers from the command `ds job-list`.

**Note:** If you are using the template of a Region Server with shared database mode (the Region Server uses the database deployed in the Central Server), you must include the following additional parameters in the `-P` option of the `ds job-create` command:

- `ComputeDBUsername=<compute_database_username>`
- `DashboardDBUsername=<dashboard_database_username>`
- `ImageDBUsername=<image_database_username>`
- `MeteringDBUsername=<metering_database_username>`
- `NetworkDBUsername=<network_database_username>`
- `OrchestrationDBUsername=<orchestration_database_username>`
- `VolumeDBUsername=<volume_database_username>`

For example:

```
ds job-create -t 1d8ef566-4eb2-4445-bafa-28eb2461bc06 \   
-P "PVCServer=172.16.2.22;PVCServerUser=admin; \   
PVCServerPassword=passw0rd;PVCQpidPassword=passw0rd; \   
MGMNetInterface=eth0;ComputeDBUsername=nova3; \   
DashboardDBUsername=dash3;ImageDBUsername=glance3; \   
MeteringDBUsername=ceil3;NetworkDBUsername=neutron3; \   
OrchestrationDBUsername=heat3;VolumeDBUsername=cinder3" \   
-N "powervc_region_server=e76db559-2605-4e68-b69b-4de9ccdedb5; \   
neutron_network_node=d1283443-f62c-4a1c-94f7-e66cc7c90ec" \   
-p e76db559-2605-4e68-b69b-4de9ccdedb5 \   
power-region-server-neutron
```

If a problem occurs during job creation and the job status is WARNING or ERROR, run the `ds job-show <job ID>` command to inspect the details of the job which also include the related error message.

6. Execute the job to install PowerVC region server by running the following command `ds job-execute <PowerVC job id>`

**Results**

PowerVC region server has been installed.

**Post PowerVC Region Server installation tasks:**

Complete the following steps after the PowerVC region server has been installed.

**About this task**

**SSL Certificate**

**Note:**

When this certificate is missing, the following services fail and become inactive:

- `openstack-cinder-volume`
- `openstack-glance-powervc`
- `openstack-neutron-powervc`
• openstack-nova-powervc
• openstack-cinder-powervc

These services must be started again, in this order, after the certificate is available on the Region Server

Procedure
1. Copy the SSL certificate from the PowerVC server to the PowerVC region server. The location of the SSL certificate on the PowerVC server is: /etc/pki/tls/certs/powervc.crt.
2. Copy the certificate to the PowerVC region server with the following command executed on the PowerVC region server: scp root@<powervc server>:/etc/pki/tls/certs/powervc.crt /etc/pki/tls/certs/powervc.crt

Results
The certificate has been copied to the PowerVC region server.

What to do next

Network Setup:

DHCP is not supported by the PowerVC Openstack Driver. When creating the network in Horizon for use with PowerVC, ensure that the DHCP box is disabled.

Installing a z/VM Region Server
Install a z/VM Region Server in your IBM Cloud Orchestrator environment.

Before you begin

Be sure that the Deployment Service server and the Central Servers are ready in your environment. For more information, see "Installing the Deployment Service" on page 37 and "Deploying an IBM Cloud Orchestrator environment" on page 52.

You must enable the xCAT version that is provided with z/VM 6.3. Before enabling xCAT, verify that you have installed z/VM 6.3, with APAR VM65513 and its prerequisites. To enable xCAT for z/VM 6.3, follow chapters 1-4 in the Enabling z/VM for OpenStack guide, at [http://www.vm.ibm.com/sysman/openstk.html](http://www.vm.ibm.com/sysman/openstk.html).

Note: Single System Image (SSI) is not supported.


Procedure
1. Create nodes. According to the resources of the zvm_region templates, you must create the zvm_region_server and the neutron_network_node nodes. Run the following commands, for example:
   
   ```
   ds node-create -t 'IBM::SCO::Node' -p '{Address: 172.17.66.12, Port: 22, User: root, Password: passw0rd}' rs-zenv1
   ds node-create -t 'IBM::SCO::Node' -p '{Address: 172.17.66.13, Port: 22, User: root, Password: passw0rd}' ns-zenv1
   ```

2. Check the node IDs by running the following command:
   ```
   ds node-list
   ```
### 3. Check the template ID and the relates resources and parameters by running the following commands, for example:

**ds template-list**

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>7264018a-c62e-49f3-ab20-1cffe41181d</td>
<td>sco-central-servers</td>
</tr>
<tr>
<td>abbd039b-5e4e-44bc-9770-6634272fb14</td>
<td>zvm_region</td>
</tr>
</tbody>
</table>
```

**ds template-resources-list abbd039b-5e4e-44bc-9770-6634272fb14**

```
<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>run_list</th>
<th>run_order</th>
</tr>
</thead>
<tbody>
<tr>
<td>zvm_region_server</td>
<td>Existing Machine</td>
<td>role[zvm-region]</td>
<td>1</td>
</tr>
<tr>
<td>neutron_network_node</td>
<td>Existing Machine</td>
<td>role[neutron-network-node]</td>
<td>2</td>
</tr>
</tbody>
</table>
```

### 4. Make sure that the prerequisite job to install the Central Servers completed successfully by running the following command, for example:

**ds job-list**

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>status</th>
<th>pjob_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e593793-3f48-4782-ace5-3dddb37289b</td>
<td>job-central-server</td>
<td>FINISHED</td>
<td></td>
</tr>
</tbody>
</table>
```
5. Create a job to install the z/VM Region Server and the neutron server by running the following command, for example:

   ds job-create -t abbd039b-e5ee-44bc-b770-6e334272fb14
   -N 'zvm_region_server=303a5d4f-f9bf-4ed5-a1f5-5039b5b057c7;
       neutron_network_node=6f3d9608-7942-49a8-887f-8e638a6161d2'
   -P 'MGMNetInterface=eth1;DataNetInterface=eth1;ExtNetInterface=eth1;
       XcatServer=9.37.36.66'
   -p 1e593793-3f48-4782-ace5-3ddd8b37289b job-zvm-server

Note:
- If the parameter value is not the same as default, you must set the value in parameters like -P 'MGMNetInterface=eth1;DataNetInterface=eth1;ExtNetInterface=eth1'
- If you do not specify all of the mandatory parameters, the job is created by using default values for the unspecified mandatory parameters. Such jobs are created with WARNING status.
- Identify the parent job that installs the central server like -p 1e593793-3f48-4782-ace5-3ddd8b37289b.
- For more information about template parameters, see "Customizing a z/VM Region Server deployment" on page 67.

Note: If you are using the template of a Region Server with shared database mode (the Region Server uses the database deployed in the Central Server), you must include the following additional parameters in the -P option of the ds job-create command:

   ComputeDBUsername=<compute_database_username>
   DashboardDBUsername=<dashboard_database_username>
   ImageDBUsername=<image_database_username>
   MeteringDBUsername=<metering_database_username>
   NetworkDBUsername=<network_database_username>
   OrchestrationDBUsername=<orchestration_database_username>
   VolumeDBUsername=<volume_database_username>

The following output is displayed, for example:

   +--------------------------------------+--------------------+----------+
   | id | name | status             |
   +--------------------------------------+--------------------+----------+
   | 89a8a9d6-5613-45e1-ae4a-7aec59a8427b | job-zvm-server     | CREATE   |
   +--------------------------------------+--------------------+----------+
   +--------------------------------------+----------------------------+----------------------------+
   | pjob_id | created_at | updated_at |
   +--------------------------------------+----------------------------+----------------------------+
   | 1e593793-3f48-4782-ace5-3ddd8b37289b | 2014-06-12T14:53:12.800460 | |}

If a problem occurs during job creation and the job status is WARNING or ERROR, run the ds job-show <job ID> command to inspect the details of the job which also include the related error message.

6. Execute the job to install the z/VM Region Server by running the following command, for example:

   ds job-execute 89a8a9d6-5613-45e1-ae4a-7aec59a8427b
What to do next


**Restriction:** z/VM Single System Image (SSI) configuration is not supported in IBM Cloud Orchestrator V2.4.

**Customizing a z/VM Region Server deployment:**

Customize the z/VM Region Server deployment by overriding the default parameters by using the CLI or the deployment wizard.

In the following table, only the parameters related to the z/VM Region Server deployment are listed. For more information about the deployment parameters, see "Deployment parameters" on page 43.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegionName</td>
<td>RegionZVM</td>
<td>This is the region name for OpenStack. It is important only when you want to deploy a multiple region environment. No need to change it for all-in-one based template. In a multiple region environment, region name must be unique.</td>
</tr>
<tr>
<td>OSLibvirtType</td>
<td>zvm</td>
<td>The virtualization type.</td>
</tr>
<tr>
<td>ComputeDriver</td>
<td>nova.virt.zvm.ZVMDriver</td>
<td>The OpenStack hypervisor driver.</td>
</tr>
<tr>
<td>XcatUsername</td>
<td>admin</td>
<td>The user name of xCAT server which is used for REST API call.</td>
</tr>
<tr>
<td>XcatServer</td>
<td>9.37.36.66</td>
<td>The xCAT server IP on which this nova compute node operates.</td>
</tr>
<tr>
<td>XcatZhcpNodeName</td>
<td>zhcp</td>
<td>The xCat ZHCP node name in xCAT.</td>
</tr>
<tr>
<td>XcatMaster</td>
<td>xcat</td>
<td>The xCAT master node (the node name in xCAT definition).</td>
</tr>
<tr>
<td>XcatMgtIp</td>
<td>XcatMgtIp</td>
<td>The xCat management interface IP address.</td>
</tr>
<tr>
<td>XcatMgtMask</td>
<td>255.255.255.192</td>
<td>The xCat management interface netmask.</td>
</tr>
<tr>
<td>ZvmDiskPool</td>
<td>ROOTP1</td>
<td>The disk pool name from which xCAT allocates disk for the new servers.</td>
</tr>
<tr>
<td>ZvmDiskPoolType</td>
<td>ECKD</td>
<td>The disk pool type. It can be FBA or ECKD.</td>
</tr>
<tr>
<td>ZvmHost</td>
<td>tivlp57</td>
<td>The node name of the z/VM hypervisor.</td>
</tr>
<tr>
<td>VswitchMappings</td>
<td>xcatvsw1:6443;xcatvsw2:6243,5244</td>
<td>The OSA configuration for each vSwitch. These configurations are required if the vSwitch needs to connect outside of z/VM.</td>
</tr>
</tbody>
</table>
### Table 10. z/VM Region Server deployment parameters (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatabagXcatPassword</td>
<td>admin</td>
<td>The databag of xCAT password.</td>
</tr>
<tr>
<td>DatabagZLinuxRootPassword</td>
<td>password</td>
<td>The databag of root password in Linux on System z®.</td>
</tr>
<tr>
<td>DatabagMnadmin</td>
<td>mnpass</td>
<td>The databag of password for mnadmin of xCAT.</td>
</tr>
<tr>
<td>Ml2MechanismDrivers</td>
<td>zvm</td>
<td>The Openstack network ml2 mechanism drivers.</td>
</tr>
<tr>
<td>Ml2FlatNetworks</td>
<td>xcatvsw2</td>
<td>The ml2 flat networks.</td>
</tr>
</tbody>
</table>

#### System files modified by the installation procedure

The following system files and folders are modified during the IBM Cloud Orchestrator installation.

- **On Deployment Server:**
  - `/etc/group`
  - `/etc/passwd`
  - `/etc/ntp.conf`
  - `/etc/fstab`
  - `/etc/httpd/conf/httpd.conf`
  - `/etc/httpd/conf.d/`
  - `/etc/rc.local`
  - `/etc/security/limits.conf`
  - `/etc/sysconfig/iptables`
  - `/etc/sysctl.conf`
  - `/proc/sys/net/ipv4/ip_forward`

- **On Central Server 1:**
  - `/etc/group`
  - `/etc/passwd`
  - `/etc/hosts`
  - `/etc/sysconfig/iptables`
  - `/etc/security/limits.conf`
  - `/etc/sysctl.conf`

- **On Central Server 2:**
  - `/etc/group`
  - `/etc/passwd`
  - `/etc/hosts`
  - `/etc/sysconfig/iptables`
  - `/etc/security/limits.conf`
  - `/etc/sysctl.conf`

- **On Central Server 3:**
  - `/etc/hosts`
  - `/etc/sysconfig/iptables`
  - `/etc/security/limits.conf`
  - `/etc/sysctl.conf`
  - `/etc/sudoers`
  - `/etc/ntp.conf`

- **On Region Server:**
  - `/etc/group`
  - `/etc/passwd`
  - `/etc/hosts`
  - `/etc/sysconfig/iptables`
  - `/etc/security/limits.conf`
  - `/etc/sysctl.conf`

- **On System Automation Application Manager server:**
Deploying the Distributed Active-Active (High Availability) topology

A high-availability installation requires more hardware than an installation that is not highly available, and additional process steps are needed.

To deploy the Distributed Active-Active (High Availability) topology, complete the following steps:

1. **Install the Deployment Service**, as described in “Installing the Deployment Service” on page 37.
2. **Install an external database**.
   
   The Distributed Active-Active (High Availability) topology requires the installation of an external database. This topology does not support the usage of a shared database that is installed on the Central Server. This topology also does not support the usage of a local database for a Region Server. The external database instance is used to run all databases for the Central Servers and the Region Servers in the Distributed Active-Active (High Availability) topology, and enables customers to use established operations procedures from other application databases. Because the database instance is a key component of IBM Cloud Orchestrator, you should install IBM DB2 with high-availability features (for example, DB2 HADR or DB2 with shared disk). For more information, see Configuring a clustered environment using DB2 High Availability Instance Configuration Utility (db2haicu) in the IBM DB2 documentation [http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.ha.doc/doc/t0052800.html](http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.admin.ha.doc/doc/t0052800.html).
3. **Configure the external database**, as described in “Configuring an external database” on page 48.
4. **Install System Automation Application Manager**, as described in “Installing System Automation Application Manager” on page 70.
   
   System Automation Application Manager is used to control the starting, stopping, and automatic restarting of IBM Cloud Orchestrator. System Automation Application Manager controls all IBM Cloud Orchestrator application components, simplifies the daily operation tasks, and can automatically restart failed components. System Automation Application Manager is installed with its own instance of IBM DB2 installed on the same node. The separation of the IBM Cloud Orchestrator database from System Automation Application Manager is needed because System Automation Application Manager is used to control the IBM Cloud Orchestrator database instance.

   **Tip:** The System Automation Application Manager installation can be manually enhanced by configuration as a two-node high-availability cluster. For more information about this configuration, see Planning for high availability in the System Automation Application Manager documentation [https://www-01.ibm.com/support/knowledgecenter/#!/SSPQ7D_4.1.0/com.ibm.saam.doc_4.1/toplan_planinstallha.html](https://www-01.ibm.com/support/knowledgecenter/#!/SSPQ7D_4.1.0/com.ibm.saam.doc_4.1/toplan_planinstallha.html).
5. **Install the active-active Central Servers**, as described in “Installing the Central Servers” on page 71.
   
   System Automation for Multiplatforms is installed on both the Central Server 2 primary node and the Central Server 2 secondary node. System Automation for Multiplatforms is a high-availability cluster solution that is used to provide failover for components such as haproxy, IBM HTTP Server, and so on. The
automation policy for System Automation for Multiplatforms is fully automated. The cluster is configured and started after the installation.

6. Install one or more Region Servers, as described in “Installing the Region Server” on page 74.

The Distributed Active-Active (High Availability) topology supports the following Region Server configurations:

- VMware Region with Neutron network
- VMware Region with Nova network
- KVM Region with Neutron network
- KVM Region with Nova network

For all Region Server installations, a policy snippet is created automatically and is included in the overall System Automation Application Manager policy.

Restriction: For a high-availability installation, only VMware and KVM regions are supported. IBM Cloud Orchestrator V2.4 does not support the use of a high-availability Central Server installation with a PowerVC region or z/VM region.

The Distributed Active-Active (High Availability) topology is now installed. Complete the final configuration steps as described in “Configuring high availability” on page 145.

For more information about high availability, see “Managing high availability” on page 187.

Installing System Automation Application Manager

Use the HA-saam template to install System Automation Application Manager in the high-availability topology.

Procedure

1. Create a virtual machine with a supported operating system installed. For information about supported operating systems see Supported hardware and operating systems.

2. Create a node resource for the virtual machine, and register the resource in the Deployment Service:

   ```
   ds node-create -t "IBM::SCO::Node"
   -p "{Address: saam_address, Port: 22, User: root, Password: password }" saam
   ```

   where

   - **IBM::SCO::Node** indicates that the virtual machine is registered as a node resource.
   - **saam_address** is the IP address of the virtual machine.
   - **password** is the root password for the virtual machine.

3. Run the `ds template-list` command to find the ID of the HA-saam template.

4. Run the `ds node-list` command to find the ID of the saam node that you created in step 2.

5. Create the deployment job:

   ```
   ds job-create -t template_id -P "MGMNetInterface=net_interface"
   -N "saam=saam_nodeid" saam-job
   ```
where

- **template_id** is the ID of the HA-saam template, which you identified in step 3 on page 70.

- **net_interface** is the network interface that is used for communication between the IBM Cloud Orchestrator management components: for example, eth0. This value must be consistent on each node.

- **saam_nodeid** is the ID of the saam node, which you identified in step 4 on page 70.

If a problem occurs during job creation and the job status is WARNING or ERROR, run the `ds job-show <job ID>` command to inspect the details of the job which also include the related error message.

6. Run the deployment job:
   ```
   ds job-execute saam-job
   ```

**Results**

System Automation Application Manager is installed in the high-availability topology.

**Installing the Central Servers**

Use the HA-sco-central-servers-extdb template to install the Central Servers in the high-availability topology.

**Before you begin**

The high availability setup requires the usage of an external database.

For more information about how to create the external databases and how to register the database resources, see "Configuring an external database" on page 48.

**Procedure**

1. Create four virtual machines with a supported operating system installed, to be configured as the Central Servers: Central Server 1, Central Server 2 primary, Central Server 2 secondary, and Central Server 3.

2. Create a node resource for each virtual machine, and register the resources in the Deployment Service:
   ```
   ds node-create -t "IBM::SCO::Node"
   -p "{{Address: node_address, Port: port, User: user_name, Password: password}}"
   central_server_1
   
   ds node-create -t "IBM::SCO::Node"
   -p "{{Address: node_address, Port: port, User: user_name, Password: password}}"
   central_server_2p
   
   ds node-create -t "IBM::SCO::Node"
   -p "{{Address: node_address, Port: port, User: user_name, Password: password}}"
   central_server_2s
   ```
ds node-create -t "IBM::SCO::Node"
   -p "{Address: node_address, Port: port,
        User: user_name, Password: password}"
central_server3

where

*IBM::SCO::Node*

indicates that the virtual machine is registered as a node resource.

*node_address*

is the IP address of the virtual machine.

*port*

is the port number to access the virtual machine.

*user_name*

is the name of a root user on the virtual machine.

*password*

is the password for the specified root user on the virtual machine.

3. Run the `ds template-list` command to find the ID of the HA-sco-central-servers-extdb template.

4. Reregister the saam node installed in “Installing System Automation Application Manager” on page 70 with a different name, for example, cs_saam. Run `ds node-create -t "IBM::SCO::Node" -p "{Address: saam_address, Port: 22, User: root, Password: password }" cs_saam`. Use the nodeid created there in the subsequent `ds job-create` command.

5. Run the `ds node-list` command to find the IDs of the following resources:
   - The System Automation Application Manager resource: saam
   - The Central Server resources that you created in step 2 on page 71: central_server_1, central_server_2_p, central_server_2_s, and central_server_3
   - The external database resources: bpm_db, cmn_db, dashboard_db, keystone_db, metering_db, and pdw_db.

6. Create the deployment job. The parameters within the command must be entered without any spaces. The parameters are shown on separate lines for clarity.

   `job-create -t <template_id>
      -P "CentralServer2VirtualAddress=<CentralServer2VirtualAddress>;
          CentralServer2VirtualHostname=<CentralServer2VirtualHostname>;
          CentralServer2VirtualNetmask=<CentralServer2VirtualNetmask>;
          CentralServer2VirtualTieBreaker=<CentralServer2VirtualTieBreaker>;
          MGMNetInterface=<net_interface>;
          SingleSignOnDomain=<SingleSignOnDomain>"
      -N "saam=<saam_nodeid>;
          central_server_1=<central_server_1_nodeid>;
          central_server_2_p=<central_server_2_p_nodeid>;
          central_server_2_s=<central_server_2_s_nodeid>;
          central_server_3=<central_server_3_nodeid>;
          bpm_db=<bpm_db_nodeid>;
          dashboard_db=<dashboard_db_nodeid>;
          keystone_db=<keystone_db_nodeid>;
          metering_db=<metering_db_nodeid>;
          pdw_db=<pdw_db_nodeid>;
          cmn_db=<cmn_db_nodeid>"
      HA-sco-central-servers-extdb_job`

where
7. Run the deployment job:

```
    ds job-execute HA-sco-central-servers-extdb_job
```

## Results

The Central Servers are installed in the high-availability topology.

### Related concepts:

- [“Using the command-line interface to deploy IBM Cloud Orchestrator” on page 54](#)
  
  You can use the command-line interface to deploy your IBM Cloud Orchestrator environment.

- [“Deployment parameters” on page 43](#)
  
  Check the list of all deployment parameters that you can configure before deploying IBM Cloud Orchestrator, and the parameter default values.

- [“Configuring an external database” on page 48](#)
  
  You can configure an external database server to be used by IBM Cloud Orchestrator.
Installing the Region Server

Use one of the templates provided to install the Region Server in the high-availability topology. You can install a KVM region or VMware region, and each region can have Neutron network.

Before you begin

Before you start to install a Region Server, you must create the OpenStack database on your external IBM® DB2® installation. You must also create the following database resources, and register the database resources in the Deployment Service:

- compute_db
- image_db
- network_db
- orchestration_db
- volume_db

Note: compute_db uses nova user, image_db uses glance user, network_db uses neutron user, orchestration_db uses heat user and volume_db uses cinder user.

For more information about how to create the external database and how to register the database resources, see “Configuring an external database” on page 48.

Related concepts:

“Using the command-line interface to deploy IBM Cloud Orchestrator” on page 54
You can use the command-line interface to deploy your IBM Cloud Orchestrator environment.

“Deployment parameters” on page 43
Check the list of all deployment parameters that you can configure before deploying IBM Cloud Orchestrator, and the parameter default values.

Neutron network topology and configurable template parameters
This topic describes the Neutron network topology.

“Configuring an external database” on page 48
You can configure an external database server to be used by IBM Cloud Orchestrator.

Installing a KVM Region Server with Neutron network

Use the HA-kvm_region-with-compute-neutron-extdb template to install a KVM Region Server with Neutron network in the high-availability topology.

Before you begin

Create the external databases and register the database resources, as described in “Installing the Region Server.”

Procedure

1. Create four virtual machines with a supported operating system installed, to be configured as the Region Server: KVM Region Server primary, KVM Region Server secondary, KVM compute node, and Neutron network node.

2. Create a node resource for each virtual machine, and register the resources in the Deployment Service:

```bash
ds node-create -t "IBM::SCO::Node"
    -p "{Address: node_address, Port: port,
        User: user_name, Password: password }"
kvm_region_server_p
```
ds node-create -t "IBM::SCO::Node"
   -p "{Address: node_address, Port: port,
       User: user_name, Password: password }"
kvm_region_server_s

ds node-create -t "IBM::SCO::Node"
   -p "{Address: node_address, Port: port,
       User: user_name, Password: password }"
kvm_compute

ds node-create -t "IBM::SCO::Node"
   -p "{Address: node_address, Port: port,
       User: user_name, Password: password }"
neutron_network_node

where

**IBM::SCO::Node**

indicates that the virtual machine is registered as a node resource.

**node_address**

is the IP address of the virtual machine.

**port**

is the port number to access the virtual machine.

**user_name**

is the name of a root user on the virtual machine.

**password**

is the password for the specified root user on the virtual machine.

3. Run the **ds template-list** command to find the ID of the HA-kvm_region-with-compute-neutron-extdb template.

4. Reregister the saam node installed in "Installing System Automation Application Manager" on page 70 with a different name, for example, kvm_saam. Run ds job-create -t "IBM::SCO::Node" -p "{Address: saam_address, Port: 22, User: root, Password: password }" kvm_saam. Use the nodeid created there in the subsequent ds job-create command.

5. Run the **ds node-list** command to find the IDs of the following resources:
   - The System Automation Application Manager resource: kvm_saam that you created in step 4.
   - The Region Server resources that you created in step 2 on page 74: kvm_region_server_p, kvm_region_server_s, kvm_compute, and neutron_network_node.
   - The external database resources: compute_db, image_db, network_db, orchestration_db, and volume_db.

6. Run the **ds job-list** command to find the ID of the HA-sco-central-servers-extdb_job job.

7. Create the deployment job:

```
ds job-create -t template_id
   -P "RegionServerVirtualAddress=RegionServerVirtualAddress;
       RegionServerVirtualHostname=RegionServerVirtualHostname;
       RegionServerVirtualNetmask=RegionServerVirtualNetmask;
       RegionServerVirtualTiebreaker=RegionServerVirtualTiebreaker;
       MGMNetInterface=net_interface"
   -N "saam=saam_nodeid;
       kvm_region_server_p=kvm_region_server_p_nodeid;
       kvm_region_server_s=kvm_region_server_s_nodeid;
       kvm_compute=kvm_compute_nodeid;
       neutron_network_node=neutron_network_node_nodeid;
       compute_db=compute_db_nodeid;"
```

Chapter 2. Installing 75
where

*template_id*

is the ID of the **HA-kvm_region-neutron-extdb_job** template, which you identified in step 3 on page 75.

RegionServerVirtualAddress

is the virtual address to be used for the Region Server. The virtual address is an additional IP address, which is assigned to the active virtual machine of a System Automation for Multiplatforms cluster. If the primary virtual machine fails, the virtual address is moved to the secondary virtual machine, and the secondary virtual machine becomes the active virtual machine.

RegionServerVirtualHostname

is the host name of the virtual address to be used for the Region Server.

RegionServerVirtualNetmask

is the netmask of the virtual address to be used for the Region Server.

RegionServerVirtualTieBreaker

is the TieBreaker for the System Automation for Multiplatforms cluster on the Region Server. If a cluster split occurs, the tiebreaker determines which part of the cluster gets quorum and can manage active resources. For example, you can use the gateway of the network as the tiebreaker. For more information about quorums and tiebreakers, see Operational Quorum.

net_interface

is the network interface that is used for communication between the IBM Cloud Orchestrator management components: for example, eth0. This value must be consistent on each node.

SingleSignOnDomain

is the DNS domain name where the manage-from components are installed. The manage-from components include IBM HTTP Server, IBM Cloud Orchestrator user interfaces, Business Process Manager, and Workload Deployer.

<X>_nodeid

are the resource IDs that you identified in step 5 on page 75.

central_server_job_id

is the ID of the **HA-sco-central-servers-extdb_job** job, which you identified in step 6 on page 75.

If a problem occurs during job creation and the job status is WARNING or ERROR, run the `ds job-show <job ID>` command to inspect the details of the job which also include the related error message.

8. Run the deployment job:

`ds job-execute HA-kvm_region-with-compute-neutron-extdb_job`
Results

The KVM Region Server with Neutron network is installed in the high-availability topology.

Installing a KVM Region Server with Nova network

Use the HA-kvm_region-with-compute-extdb template to install a KVM Region Server with Nova network in the high-availability topology.

Before you begin

Create the external databases and register the database resources, as described in “Installing the Region Server” on page 74.

Procedure

1. Create three virtual machines with a supported operating system installed, to be configured as the Region Server: KVM Region Server primary, KVM Region Server secondary, and KVM compute node.
2. Create a node resource for each virtual machine, and register the resources in the Deployment Service:

   \[
   \text{ds node-create -t "IBM::SCO::Node" -p \{Address: node_address, Port: port, User: user_name, Password: password \}}
   \]

   where

   IBM::SCO::Node indicates that the virtual machine is registered as a node resource.

   node_address is the IP address of the virtual machine.

   port is the port number to access the virtual machine.

   user_name is the name of a root user on the virtual machine.

   password is the password for the specified root user on the virtual machine.

3. Run the ds template-list command to find the ID of the HA-kvm_region-with-compute-extdb template.
4. Reregister the saam node installed in “Installing System Automation Application Manager” on page 70 with a different name, for example, kvm_saam. Run \[
   \text{ds node-create -t "IBM::SCO::Node" -p \{Address: saam_address, Port: 22, User: root, Password: password \}}
   \]

   Use the nodeid created there in the subsequent ds node-create command.
5. Run the ds node-list command to find the IDs of the following resources:

   - The System Automation Application Manager resource: saam
The Region Server resources that you created in step 2 on page 77:
- `kvm_region_server_p`, `kvm_region_server_s`, and `kvm_compute`

The external database resources: `compute_db`, `image_db`, `network_db`, `nova_db`,
- `orchestration_db`, and `volume_db`

6. Run the `ds job-list` command to find the ID of the HA-sco-central-servers-extdb_job job.

7. Create the deployment job:

```bash
ds job-create -t template_id
-P "RegionServerVirtualAddress=RegionServerVirtualAddress;
  RegionServerVirtualHostname=RegionServerVirtualHostname;
  RegionServerVirtualNetmask=RegionServerVirtualNetmask;
  RegionServerVirtualTieBreaker=RegionServerVirtualTieBreaker;
  MGMNetInterface=net_interface;
  SingleSignOnDomain=SingleSignOnDomain"
-N "saam=saam_nodeid;
  kvm_region_server_p=kvm_region_server_p_nodeid;
  kvm_region_server_s=kvm_region_server_s_nodeid;
  kvm_compute=kvm_compute_nodeid;
  compute_db=compute_db_nodeid;
  image_db=image_db_nodeid;
  network_db=network_db_nodeid;
  nova_db=nova_db_nodeid;
  orchestration_db=orchestration_db_nodeid;
  volume_db=volume_db_nodeid"
-p central_server_job_id
-HA-kvm_region-neutron-extdb_job
```

where

- `template_id` is the ID of the HA-kvm_region-with-compute-extdb template, which you identified in step 3 on page 77.

- `RegionServerVirtualAddress` is the virtual address to be used for the Region Server. The virtual address is an additional IP address, which is assigned to the active virtual machine of a System Automation for Multiplatforms cluster. If the primary virtual machine fails, the virtual address is moved to the secondary virtual machine, and the secondary virtual machine become the active virtual machine.

- `RegionServerVirtualHostname` is the host name of the virtual address to be used for the Region Server.

- `RegionServerVirtualNetmask` is the netmask of the virtual address to be used for the Region Server.

- `RegionServerVirtualTieBreaker` is the TieBreaker for the System Automation for Multiplatforms cluster on the Region Server. If a cluster split occurs, the tiebreaker determines which part of the cluster gets quorum and can manage active resources. For example, you can use the gateway of the network as the tiebreaker. For more information about quorums and tiebreakers, see Operational quorum.

- `net_interface` is the network interface that is used for communication between the IBM Cloud Orchestrator management components: for example, `eth0`. This value must be consistent on each node.

- `SingleSignOnDomain` is the DNS domain name where the manage-from components are
installed. The manage-from components include IBM HTTP Server, IBM Cloud Orchestrator user interfaces, Business Process Manager, and Workload Deploer.

\(<X>_nodeid\)

are the resource IDs that you identified in step 5 on page 77.

\(central_server_job_id\)

is the ID of the HA-sco-central-servers-extdb_job job, which you identified in step 6 on page 78.

If a problem occurs during job creation and the job status is WARNING or ERROR, run the \(ds\ job-show \<job\ ID\>\) command to inspect the details of the job which also include the related error message.

8. Run the deployment job:

\(ds\ job-execute HA-kvm_region-with-compute-extdb_job\)

Results

The KVM Region Server with Nova network is installed in the high-availability topology.

Installing a VMware Region Server with Neutron network

Use the HA-vmware_region-neutron-extdb template to install a VMware Region Server with Neutron network in the high-availability topology.

Before you begin

Create the external databases and register the database resources, as described in “Installing the Region Server” on page 74.

Procedure

1. Create three virtual machines with a supported operating system installed, to be configured as the Region Server: VMware Region Server primary, VMware Region Server secondary, and Neutron network node.

2. Create a node resource for each virtual machine, and register the resources in the Deployment Service:

\(ds\ node-create -t "IBM::SCO::Node" -p \{Address: node_address, Port: port, User: user_name, Password: password \} vmware_region_server_p\)

\(ds\ node-create -t "IBM::SCO::Node" -p \{Address: node_address, Port: port, User: user_name, Password: password \} vmware_region_server_s\)

\(ds\ node-create -t "IBM::SCO::Node" -p \{Address: node_address, Port: port, User: user_name, Password: password \} neutron_network_node\)

where

\(IBM::SCO::Node\)

indicates that the virtual machine is registered as a node resource.

\(node_address\)

is the IP address of the virtual machine.
port is the port number to access the virtual machine.

user_name is the name of a root user on the virtual machine.

password is the password for the specified root user on the virtual machine.

3. Run the `ds template-list` command to find the ID of the HA-vmware_region-neutron-extdb template.

4. Reregister the `saam` node installed in "Installing System Automation Application Manager" on page 70 with a different name, for example, `vmware_n_saam`. Run `ds node-create -t "IBM::SCO::Node" -p "{Address: saam_address, Port: 22, User: root, Password: password }" vmware_n_saam`. Use the nodeid created there in the subsequent `ds node-create` command.

5. Run the `ds node-list` command to find the IDs of the following resources:
   - The System Automation Application Manager resource: `vmware_n_saam` as specified in the previous step.
   - The Region Server resources that you created in step 2 on page 79: `vmware_region_server_p`, `vmware_region_server_s`, and `neutron_network_node`
   - The external database resources: `compute_db`, `image_db`, `network_db`, `nova_db`, `orchestration_db`, and `volume_db`

6. Run the `ds job-list` command to find the ID of the HA-sco-central-servers-extdb_job job.

7. Create the deployment job:
   ```
   ds job-create -t template_id
   -P "RegionServerVirtualAddress=RegionServerVirtualAddress;
   RegionServerVirtualHostname=RegionServerVirtualHostname;
   RegionServerVirtualNetmask=RegionServerVirtualNetmask;
   RegionServerVirtualTieBreaker=RegionServerVirtualTieBreaker;
   VMServerHost=VMServerHost;
   VMServerUserName=VMServerUserName;
   VMServerPassword=VMServerPassword;
   VMClusterName=VMClusterName;
   MGMNetInterface=net_interface"
   -N "saam=saam_nodeid;
   vmware_region_server_p=vmware_region_server_p_nodeid;
   vmware_region_server_s=vmware_region_server_s_nodeid;
   neutron_network_node=neutron_network_node_nodeid;
   compute_db=compute_db_nodeid;
   image_db=image_db_nodeid;
   network_db=network_db_nodeid;
   orchestration_db=orchestration_db_nodeid;
   volume_db=volume_db_nodeid"
   -p central_server_job_id
   HA-vmware_region-neutron-extdb_job
   ```

   where

   template_id is the ID of the HA-vmware_region-neutron-extdb template, which you identified in step 3.

   `RegionServerVirtualAddress` is the virtual address to be used for the Region Server. The virtual address is an additional IP address, which is assigned to the active virtual machine of a System Automation for Multiplatforms cluster. If
the primary virtual machine fails, the virtual address is moved to the
secondary virtual machine, and the secondary virtual machine become
the active virtual machine.

RegionServerVirtualHostname
is the host name of the virtual address to be used for the Region Server.

RegionServerVirtualNetmask
is the netmask of the virtual address to be used for the Region Server.

RegionServerVirtualTieBreaker
is the TieBreaker for the System Automation for Multiplatforms cluster
on the Region Server. If a cluster split occurs, the tiebreaker determines
which part of the cluster gets quorum and can manage active resources.
For example, you can use the gateway of the network as the tiebreaker.
For more information about quorums and tiebreakers, see Operational
quorum

VMServerHost
is the IP address of the vCenter server.

VMServerUserName
is the name of a user on the vCenter server.

VMServerPassword
is the password for the specified user on the vCenter server.

VMClusterName
is the cluster name in vCenter that is used to start the virtual machine.

net_interface
is the network interface that is used for communication between the
IBM Cloud Orchestrator management components: for example, eth0.
This value must be consistent on each node.

<X>_nodeid
are the resource IDs that you identified in step 5 on page 80.

central_server_job_id
is the ID of the HA-sco-central-servers-extdb_job job, which you
identified in step 6 on page 80.

If a problem occurs during job creation and the job status is WARNING or ERROR,
run the ds job-show <job ID> command to inspect the details of the job which
also include the related error message.

8. Run the deployment job:
   ds job-execute HA-vmware_region-neutron-extdb_job

Results

The VMware Region Server with Neutron network is installed in the
high-availability topology.
Installing a VMware Region Server with Nova network

Use the HA-vmware_region-extdb template to install a VMware Region Server with Nova network in the high-availability topology.

Before you begin

Create the external databases and register the database resources, as described in “Installing the Region Server” on page 74.

Procedure

1. Create two virtual machines with a supported operating system installed, to be configured as the Region Server: VMware Region Server primary and VMware Region Server secondary.

2. Create a node resource for each virtual machine, and register the resources in the Deployment Service:

   ```
   ds node-create -t "IBM::SCO::Node"
   -p "{Address: node_address, Port: port, User: user_name, Password: password }"
   vmware_region_server_p
   ds node-create -t "IBM::SCO::Node"
   -p "{Address: node_address, Port: port, User: user_name, Password: password }"
   vmware_region_server_s
   ```

   where

   - **IBM::SCO::Node** indicates that the virtual machine is registered as a node resource.
   - **node_address** is the IP address of the virtual machine.
   - **port** is the port number to access the virtual machine.
   - **user_name** is the name of a root user on the virtual machine.
   - **password** is the password for the specified root user on the virtual machine.

3. Run the `ds template-list` command to find the ID of the HA-vmware_region-extdb template.

4. Reregister the saam node installed in “Installing System Automation Application Manager” on page 70 with a different name, for example, `vmware_saam`. Run `ds node-create -t "IBM::SCO::Node" -p "{Address: saam_address, Port: 22, User: root, Password: password }" vmware_saam`. Use the nodeid created there in the subsequent `ds node-create` command.

5. Run the `ds node-list` command to find the IDs of the following resources:
   - The System Automation Application Manager resource: `saam`
   - The Region Server resources that you created in step 2: `vmware_region_server_p` and `vmware_region_server_s`
   - The external database resources: `compute_db`, `image_db`, `network_db`, `nova_db`, `orchestration_db`, and `volume_db`

6. Run the `ds job-list` command to find the ID of the HA-sco-central-servers-extdb_job job.

7. Create the deployment job:
ds job-create -t template_id
   -P "RegionServerVirtualAddress=RegionServerVirtualAddress;
   RegionServerVirtualHostname=RegionServerVirtualHostname;
   RegionServerVirtualNetmask=RegionServerVirtualNetmask;
   RegionServerVirtualTieBreaker=RegionServerVirtualTieBreaker;
   VMServerHost=VMServerHost;
   VMServerUsername=VMServerUsername;
   VMServerPassword=VMServerPassword;
   VMClusterName=VMClusterName;
   MGNetInterface=net_interface;
   SingleSignOnDomain=SingleSignOnDomain"
   -N "saam=saam_nodeid;
   vmware_region_server_p=vmware_region_server_p_nodeid;
   vmware_region_server_s=vmware_region_server_s_nodeid;
   compute_db=compute_db_nodeid;
   image_db=image_db_nodeid;
   network_db=network_db_nodeid;
   nova_db=nova_db_nodeid;
   orchestration_db=orchestration_db_nodeid;
   volume_db=volume_db_nodeid"
   -p central_server_job_id
HA-kvm_region-neutron-extdb_job

where

**template_id**

is the ID of the HA-vmware_region-extdb template, which you identified in step 3 on page 82.

**RegionServerVirtualAddress**

is the virtual address to be used for the Region Server. The virtual address is an additional IP address, which is assigned to the active virtual machine of a System Automation for Multiplatforms cluster. If the primary virtual machine fails, the virtual address is moved to the secondary virtual machine, and the secondary virtual machine becomes the active virtual machine.

**RegionServerVirtualHostname**

is the host name of the virtual address to be used for the Region Server.

**RegionServerVirtualNetmask**

is the netmask of the virtual address to be used for the Region Server.

**RegionServerVirtualTieBreaker**

is the TieBreaker for the System Automation for Multiplatforms cluster on the Region Server. If a cluster split occurs, the tiebreaker determines which part of the cluster gets quorum and can manage active resources. For example, you can use the gateway of the network as the tiebreaker. For more information about quorums and tiebreakers, see Operational quorum.

**VMServerHost**

is the IP address of the vCenter server.

**VMServerUsername**

is the name of a user on the vCenter server.

**VMServerPassword**

is the password for the specified user on the vCenter server.

**VMClusterName**

is the cluster name in vCenter that is used to start the virtual machine.

**net_interface**

is the network interface that is used for communication between the
IBM Cloud Orchestrator management components: for example, eth0. This value must be consistent on each node.

SingleSignOnDomain

is the DNS domain name where the manage-from components are installed. The manage-from components include IBM HTTP Server, IBM Cloud Orchestrator user interfaces, Business Process Manager, and Workload Deployer.

<X>_nodeid

are the resource IDs that you identified in step 5 on page 82.

central_server_job_id

is the ID of the HA-sco-central-servers-extdb_job job, which you identified in step 6 on page 82.

If a problem occurs during job creation and the job status is WARNING or ERROR, run the ds job-show <job ID> command to inspect the details of the job which also include the related error message.

8. Run the deployment job:

ds job-execute HA-vmware_region-extdb_job

Results

The VMware Region Server with Nova network is installed in the high-availability topology.

Verifying the installation

When you have completed the installation, you can verify it by completing the following verification steps.

Procedure

1. Access the IBM Cloud Orchestrator user interfaces:

To access each user interface, use the administrator user name and the IBM Cloud Orchestrator password that you specified with the OrchestratorPassword parameter in the deployment template. The Default value is used as domain name.

For information about the IBM Cloud Orchestrator user interfaces, see “Accessing IBM Cloud Orchestrator user interfaces” on page 174.

2. Navigate the self service offerings in IBM Cloud Orchestrator UI.

3. Verify that the offerings and categories have been created successfully:

   a. Go to CONFIGURATION > Categories and verify the following categories:

      • Virtual System Operations (Single Instance)
      • Virtual System Operations (Multiple Instances)

   b. Go to CONFIGURATION > Offerings and verify the following categories:

      • Deploy Virtual System Instance
      • Start Virtual System Instance
      • Stop Virtual System Instance
      • Restart Virtual System Instance
      • Delete Virtual System Instance
      • Change End Time of Virtual System Instance
      • Change Limits of Environmental Profile
- Start Single Server Virtual System Instance
- Stop Single Server Virtual System Instance
- Restart Single Server Virtual System Instance
- Delete Single Server Virtual System Instance
- Modify Single Server Virtual System Instance
- Start Multiple Virtual System Instanvices
- Stop Multiple Virtual System Instanvices
- Restart Multiple Virtual System Instanvices
- Delete Multiple Virtual System Instanvices

4. Verify the connectivity to the KVM Compute Node or VMware vCenter or VMControl, by using the OpenStack command line.

To configure and administer IBM Cloud Orchestrator, you might need to use the Keystone command-line interface to manage authentication and authorizations, and the Glance command-line interface to manage virtual images. The environment profile /root/keystonerc is generated on each Region Server. Make sure that the OpenStack command-line interface can be executed by using the profile, as shown in the following steps. Run the following steps on every Region Server. If you installed IBM Cloud Orchestrator as highly available, execute the steps on each primary Region Server.

a. Run the following command to invoke the profile:
   ```
   source /root/openrc
   ```

b. Run the following command to display a list of services:
   ```
   nova service-list
   ```

The output should be displayed in the following format. If the environment is working, the state of each service is :):

- **For a KVM Region:**

<table>
<thead>
<tr>
<th>Binary</th>
<th>Host</th>
<th>Zone</th>
<th>Status</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova-conductor</td>
<td>kvm-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
<tr>
<td>nova-scheduler</td>
<td>kvm-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
<tr>
<td>nova-consoleauth</td>
<td>kvm-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
<tr>
<td>nova-cert</td>
<td>kvm-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>down</td>
</tr>
<tr>
<td>nova-compute</td>
<td>europe1-kvm24-2</td>
<td>nova</td>
<td>enabled</td>
<td>up</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Updated_at</th>
<th>Disabled Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-07-23T10:27:21.903063</td>
<td>-</td>
</tr>
<tr>
<td>2014-07-23T10:27:13.164215</td>
<td>-</td>
</tr>
<tr>
<td>2014-07-18T12:39:05.980178</td>
<td>-</td>
</tr>
<tr>
<td>2014-07-23T10:21.927675</td>
<td>-</td>
</tr>
<tr>
<td>2014-07-23T10:19.360913</td>
<td>-</td>
</tr>
</tbody>
</table>

- **For a VMware Region:**

<table>
<thead>
<tr>
<th>Binary</th>
<th>Host</th>
<th>Zone</th>
<th>Status</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova-conductor</td>
<td>vmware-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
<tr>
<td>nova-scheduler</td>
<td>vmware-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
<tr>
<td>nova-consoleauth</td>
<td>vmware-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
<tr>
<td>nova-cert</td>
<td>vmware-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
<tr>
<td>nova-compute</td>
<td>vmware-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
<tr>
<td>nova-console</td>
<td>vmware-region-europe1</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Updated_at</th>
<th>Disabled Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-07-23T10:15:05.290178</td>
<td>-</td>
</tr>
<tr>
<td>2014-07-23T10:13.164215</td>
<td>-</td>
</tr>
<tr>
<td>2014-07-18T12:39:05.980178</td>
<td>-</td>
</tr>
<tr>
<td>2014-07-23T10:21.927675</td>
<td>-</td>
</tr>
<tr>
<td>2014-07-23T10:19.360913</td>
<td>-</td>
</tr>
</tbody>
</table>
For more information about the OpenStack command-line interface, refer to the [OpenStack User Guide](#).

c. Check the hypervisor status by running the `nova hypervisor-list` command.

Get the ID of the hypervisor and run the `nova hypervisor-show <id>` command.

- **For a KVM Region:**

<table>
<thead>
<tr>
<th>ID</th>
<th>Hypervisor hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>europe1-kvm24-2.24.customer.ibm.com</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu_info_arch</td>
<td>x86_64</td>
</tr>
<tr>
<td>cpu_info_features</td>
<td>['rdtsdp', 'dca', 'pdcm', 'xtop', 'tmz', 'est', 'vmx', 'ds_col', 'monitor', 'dtes64', 'pbe', 'tm', 'ht', 'ss', 'acpi', 'ds', 'vme']</td>
</tr>
<tr>
<td>cpu_info_model</td>
<td>Nehalem</td>
</tr>
<tr>
<td>cpu_info_topology_cores</td>
<td>4</td>
</tr>
<tr>
<td>cpu_info_topology_sockets</td>
<td>1</td>
</tr>
<tr>
<td>cpu_info_topology_threads</td>
<td>2</td>
</tr>
<tr>
<td>cpu_info_vendor</td>
<td>Intel</td>
</tr>
<tr>
<td>current_workload</td>
<td>0</td>
</tr>
<tr>
<td>disk_available_least</td>
<td>-29</td>
</tr>
<tr>
<td>free_disk_gb</td>
<td>-11</td>
</tr>
<tr>
<td>free_ram_mb</td>
<td>41600</td>
</tr>
<tr>
<td>host_ip</td>
<td>172.16.24.2</td>
</tr>
<tr>
<td>hypervisor_hostname</td>
<td>europe1-kvm24-2.24.customer.ibm.com</td>
</tr>
<tr>
<td>hypervisor_type</td>
<td>QEMU</td>
</tr>
<tr>
<td>hypervisor_version</td>
<td>12001</td>
</tr>
<tr>
<td>id</td>
<td>1</td>
</tr>
<tr>
<td>local_gb</td>
<td>49</td>
</tr>
<tr>
<td>local_gb_used</td>
<td>60</td>
</tr>
<tr>
<td>memory_mb</td>
<td>48256</td>
</tr>
<tr>
<td>memory_mb_used</td>
<td>6656</td>
</tr>
<tr>
<td>running_vms</td>
<td>3</td>
</tr>
<tr>
<td>service_host</td>
<td>europe1-kvm24-2</td>
</tr>
<tr>
<td>service_id</td>
<td>8</td>
</tr>
<tr>
<td>vcpus</td>
<td>16</td>
</tr>
<tr>
<td>vcpu_used</td>
<td>3</td>
</tr>
</tbody>
</table>

- **For a VMware region:**

<table>
<thead>
<tr>
<th>ID</th>
<th>Hypervisor hostname</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>domain-c7(europe-1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu_info_model</td>
<td>['Intel(R) Xeon(R) CPU X5570 @ 2.93GHz', 'Intel(R) Xeon(R) CPU X5570 @ 2.93GHz']</td>
</tr>
<tr>
<td>cpu_info_topology_cores</td>
<td>16</td>
</tr>
<tr>
<td>cpu_info_topology_threads</td>
<td>32</td>
</tr>
<tr>
<td>cpu_info_vendor</td>
<td>['IBM', 'IBM']</td>
</tr>
<tr>
<td>current_workload</td>
<td>0</td>
</tr>
<tr>
<td>disk_available_least</td>
<td>-</td>
</tr>
<tr>
<td>free_disk_gb</td>
<td>679</td>
</tr>
<tr>
<td>free_ram_mb</td>
<td>60755</td>
</tr>
<tr>
<td>host_ip</td>
<td>172.16.82.16</td>
</tr>
<tr>
<td>hypervisor_hostname</td>
<td>domain-c7(europe-1)</td>
</tr>
<tr>
<td>hypervisor_type</td>
<td>VMware vCenter Server</td>
</tr>
<tr>
<td>hypervisor_version</td>
<td>5001000</td>
</tr>
<tr>
<td>id</td>
<td>1</td>
</tr>
<tr>
<td>local_gb</td>
<td>959</td>
</tr>
</tbody>
</table>
5. View the status of the IBM Cloud Orchestrator components:
   - If you installed IBM Cloud Orchestrator as highly available, log on to System Automation Application Manager interface and verify that the root resource named Cloud Orchestrator shows the following states:
    Compound State OK
     Observed State Online
     Desired State Online
     Automated Yes

   **Note:** Refer to "Using System Automation Application Manager" on page 190 for more information on how to use System Automation Application Manager.

   - If you installed IBM Cloud Orchestrator non-highly available, view the status of the IBM Cloud Orchestrator components by using SCOrchestrator.py:
     ```
     ./SCOrchestrator.py --status
     ```

   **Note:** Before running the SCOrchestrator.py script, see "Managing services with SCOrchestrator.py" on page 175 and ensure to meet the requirements.

   The status of all components is shown as result output:
   ```
   >>> Collecting Status for IBM Cloud Orchestrator
   >>> Please wait ======>>>>>>
   Component Hostname Status
   -------------------------------
   bpm-dmgr 9.115.78.84 online
   bpm-node 9.115.78.84 online
   bpm-server 9.115.78.84 online
   db2 9.115.78.83 online
   lwd 9.115.78.87 online
   openstack-ceilometer-api 9.115.78.84 online
   openstack-ceilometer-api 9.115.78.83 online
   openstack-ceilometer-central 9.115.78.83 online
   openstack-ceilometer-collector 9.115.78.83 online
   openstack-keystone 9.115.78.84 online
   pcg 9.115.78.84 online
   qpidd 9.115.78.83 online
   swi 9.115.78.84 online
   >>> Status IBM Cloud Orchestrator complete
   ```

6. Verify the infrastructure status by navigating the Administrator user interface to check that you have the availability zones created, for example by clicking Admin > System Panel > Availability Zones. If you have a multi-region environment you can switch among regions using the drop-down list at the top right of the window. The regions you see in the Administrator user interface are different from the ones listed in the output of the keystone endpoint list because the keystone command lists also RegionCentral that is an OpenStack region used by ceilometer only.

7. Create an image, register, and then deploy it. For information about this task, see Chapter 6, "Managing virtual images," on page 325.
Troubleshooting installation

Learn how to troubleshoot installation problems.

To troubleshoot problems that occurred during the installation procedure, you can perform the following actions:

- **Check the log files.**
  
  For the Deployment Service, the log files are located in the `/var/log/cloud-deployer` directory.
  
  To do the problem determination for the *ds* command, refer to the log files located in the `/var/log/ds` directory.

  **ds-api.log**

  This log is for *ds-api* service.

  **ds-engine.log**

  This log is for *ds-engine* service.

  **ds-engine-<job_id>.log**

  This log is generated when running the *ds job-execute* command. The knife command outputs are recorded in this log file for each job.

  If any problem occurs when running the *chef bootstrap* command, check the `/var/chef/cache/chef-stacktrace.out` log file in the target system.

- **Check error details for job execution**

  Use the *ds job-show <JOB_ID>* command to show the error details in the fault section. For example:

  ```
  ds job-show aa0eb97-5d83-4acb-9589-dc8b654a6d91
  ```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>created_at</td>
<td>2014-03-27 05:47:38.271874</td>
</tr>
<tr>
<td>deleted_at</td>
<td>None</td>
</tr>
<tr>
<td>fault</td>
<td>{</td>
</tr>
</tbody>
</table>
  | | "message": "Failed to execute task from compose-topo-roles : create chef roles for deployment",
  | | "exception": |
  | | "Traceback (most recent call last):
  | | 
  | | " File "/usr/lib/python2.6/site-packages/ds/engine/deploy/task_runner.py", line 70, in execute
  | | res = task.execute(self.context, logger)"
  | | " File "/usr/lib/python2.6/site-packages/ds/engine/deploy/common/compose_topo_roles.py", line 77, in execute
  | | res_name, resource)"
  | | " File "/usr/lib/python2.6/site-packages/ds/engine/deploy/resource_handler/node_handler.py", line 14, in parse
  | | usr = res["Properties"]
  | | "KeyError: 'User'"
  | |
  |
  | heat_meta |
  | params": null,
  | "template": |
  | "AWSTemplateFormatVersion": "2010-09-09",
  | "Outputs": |
  | "ORCHESTRATION_DB_ADDR": |
  | "Value": |
  | "Fn::GetAtt": |
  | "control",
  | "PublicIp" |

**Known installation errors**

- Errors of 32-bit library files *libstdc++.so.6* and *libpam.so* were not found in *db2prereqcheck.log*
The following errors are not critical and can be ignored. The root cause is those 32-bit library files that cannot be found on the 64-bit operating system platform:

Validating "32 bit version of "libstdc++.so.6" " ...

Found the 64 bit "/usr/lib64/libstdc++.so.6" in the following directory "/usr/lib64".

DBT3514W The db2prereqcheck utility failed to find the following 32-bit library file: "libstdc++.so.6".

Validating "/lib/libpam.so*" ...

DBT3514W The db2prereqcheck utility failed to find the following 32-bit library file: "/lib/libpam.so*".

Validating "32 bit version of "libstdc++.so.6" " ...

DBT3514W The db2prereqcheck utility failed to find the following 32-bit library file: "libstdc++.so.6".

WARNING : Requirement not matched.

Requirement not matched for DB2 database "Server". Version: "10.5.0.2".
Summary of prerequisites that are not met on the current system:

DBT3514W The db2prereqcheck utility failed to find the following 32-bit library file: "/lib/libpam.so*".

• Fail to start Compute Service when using non-administrator account to connect vCenter in VMware region

If the nova Compute Service cannot be started when using the non-administrator account to connect vCenter in VMware region and has the following errors in compute.log file, you must check if the account has the minimum permissions that is required by the OpenStack integration with vCenter, for the detail privileges list you can refer to the section VMware vCenter service account in the OpenStack documentation at http://docs.openstack.org/trunk/config-reference/content/vmware.html.

2014-07-07 16:04:34.049 7390 DEBUG nova.virt.vmwareapi.driver [req-7563f6b2-d9c7-4165-9898-d8813e271015 None None] Server raised fault: 'Permission to perform this operation was denied.'

_create_session /usr/lib/python2.6/site-packages/nova/virt/vmwareapi/driver.py:995

2014-07-07 16:04:34.050 7390 DEBUG nova.virt.vmwareapi.driver [req-7563f6b2-d9c7-4165-9898-d8813e271015 None None] _call_method(session=5274632b-4c09-87ce-f2be-bf2659e02c32) failed. Module: <module 'nova.virt.vmwareapi.vim_util' from '/usr/lib/python2.6/site-packages/nova/virt/vmwareapi/vim_util.py' Method: get_inner_objects. args: [VM Object, val]}

value = {'resgroup-119': type = 'ResourcePool'}, 'vm', 'VirtualMachine', ['name', 'runtime.connectionState']]. kwargs: {}.

Iteration: 11. Exception: Error(s) NotAuthenticated occurred in the call to RetrievePropertiesEx. _call_method /usr/lib/python2.6/site-packages/nova/virt/vmwareapi/driver.py:1099

• Cannot find attached volume after volume-attach

The nova volume-attach can be used to attach a volume to an instance. Sometimes, the volume-attach command runs successfully, but when you ran the fdisk -1, you cannot find the attached volume. After you restart the virtual machine, the volume can be found. It is an known issue for VMware hosted system. Many workaround have been used to discover the attached volume
without reboot the guest operating system, such as login the guest operating system and run the following command:

```bash
  echo "- - -" > /sys/class/scsi_host/host#/scan
```

**Failed to install IBM Cloud Orchestrator because the node time is not sync with deployment service node.**

Deployment service will sync the time with the node that used to deploy IBM Cloud Orchestrator, if the time is different, it will use ntp to sync the time. But in some case if the ntp service is stopped in deployment service or the network is not accessible, the synchronization will fail and the installation will fail as well. You will see error messages in `/var/log/ds-engine-<job-id>.log` like following:

```
14 Jul 21:18:47 ntpdate[3645]: no server suitable for synchronization found
Starting ntpd:
[60G][0;32m OK [0;39m]
ip6tables: Saving firewall rules to /etc/sysconfig/ip6tables:
[60G][0;32m OK [0;39m]
Starting Chef Client, version 11.6.2
Creating a new client identity for central_server_1-zpvlv3x6m4v6 using the validator key.
```

You can manually run `ntpdate` to verify if the time service is wrong in deployment service by performing the following steps:

1. ssh to the failed node.
2. Check the `/etc/ntp.conf`. Find the last line like "server <deployment service node ip> burst iburst prefer"
3. Stop the ntpd if it is running:
   ```bash
   /etc/init.d/ntpd stop
   ```
4. Check if the time sync is able to success:
   ```bash
   ntpdate <deployment service node ip>
   ```
5. You will see the message like:
   ```
   30 Jul 04:15:46 ntpdate[19267]: adjust time server 172.16.42.55 offset -0.000122 sec
   ```
   if you see the message like:
   ```
   30 Jul 04:18:33 ntpdate[19273]: no server suitable for synchronization found
   ```
   check if ntpd is started in the deployment service node and the network is accessible, there is no iptables that is blocking the UDP port 123.

**IBM Cloud Orchestrator deployment requires the domain name used in cookies as one of the parameters of the deployment.**

Cookies that are used to implement user interface security features can be set only for domain names that are not top-level domain names and adhere to the public suffix list.

To resolve this problem, provide a domain name matching the requirements to be used in cookies as one of the parameters of the deployment.

**During deployment service installation, you break the installation manually or power off the deployment service node.**

When you try to reinstall the deployment service, the following error occurs.
Errors in the `deploy.log` file:
Resolution:
You must reinstall the operation system of deployment service node or restore the snapshot back, and reinstall deployment service.

• Deployment service installed failed with user provided repository:

Scenario:
The deployment service installation failed, and found information in /var/log/ds/deployer.log like following:

You could try using --skip-broken to work around the problem
You could try running: rpm -Va --nofiles --nodigest
This often means that the root cause is something else and multilib version checking is just pointing out that there is a problem. For example:

You have an upgrade for gnutls which is missing some dependency that another package requires. Yum is trying to solve this by installing an older version of gnutls of the different architecture. If you exclude the bad architecture

Chapter 2. Installing 91
All-in-one deployment failed

All-in-one IBM Cloud Orchestrator deployment failed with error Failed to create network.

Reason:

If you provided customized parameters when deploying the all-in-one topology of IBM Cloud Orchestrator, the parameters are passed directly to OpenStack. If the parameters are not valid, the network creation will fail. The following error is displayed in the /var/log/ds/ds-engine-<your job id>.log file:

```
[2014-08-28 04:13:46,467] 208.43.88.157 Mixlib::ShellOut::ShellCommandFailed
[2014-08-28 04:13:46,467] 208.43.88.157 ------------------------------------
[2014-08-28 04:13:46,468] 208.43.88.157 Expected process to exit with [0], but received '1'
[2014-08-28 04:13:46,469] 208.43.88.157 STDOUT:
[2014-08-28 04:13:46,470] 208.43.88.157 ---- End output of nova-manage network create --label='VM Network' --fixed_range_v4=10.32.17.224/29 --num_networks=1 --network_size=16 --bridge=br4096 --dns1=--dns2= --bridge_interface=eth1 --vlan=100

Solution:

Run the nova-manage command again on the target node and get the real error messages:
It shows ValueError: "The network range is not big enough to fit 1 networks. Network size is 16."

This is because the network defined 10.32.17.224/29 (the network size is 8) is not able to split to 1 number with size 16. In this case, you must change the parameter and redeploy IBM Cloud Orchestrator.

**Preparing to install additional Enterprise Edition components**

A complete deployment of IBM Cloud Orchestrator Enterprise Edition involves the installation of various optional components that comprise the IBM Cloud Orchestrator Enterprise Edition release.

These components include:
- Jazz for Service Management
- IBM SmartCloud Cost Management
- IBM Tivoli Monitoring

These components can be installed on physical or virtual machines, subject to the relevant hardware and software requirements. For more information about these components, visit the links below:

1. Quick Start Guide for [Jazz for Service Management](#)  
2. Quick Start Guide for [IBM SmartCloud Cost Management](#)  
3. Quick Start Guide for [IBM Tivoli Monitoring](#) and [Tivoli Monitoring for Virtual Environments](#)

**Quick start guide for metering and billing**

Use this topic as a start guide when configuring SmartCloud Cost Management for metering and billing.

The following list provides information about configuration steps required for metering and billing:

**Install Tivoli Common Reporting 3.1.0.1**  
For information about this task, see the [Installing Tivoli Common Reporting section](#) in the Jazz for Service Management information center.

**Install SmartCloud Cost Management**  
For information about this task, see [Installing SmartCloud Cost Management 2.1.0.4](#).

**Configuration required for metering**  
For more information about the configuration required for metering, see the [Automated configuration topic](#).

**Configure job processing**

- For information about configuring processing paths, see [setting processing options topic](#).
- For information about defining rates and rate templates, see [Administering the system](#).
- For information about customizing jobs, see [Administering data processing](#).
Performing post-installation tasks

After you installed IBM Cloud Orchestrator, there are additional configuration steps and management tasks that you can perform.

Assigning zones to domains and projects

Assign a zone to a domain and to a project after you complete the installation of IBM Cloud Orchestrator.

To assign a zone, you must log in to the Administration user interface as Cloud Administrator. Follow the steps described in “Assigning a zone to a domain” on page 219 and “Assigning a zone to a project” on page 224.

Setting NTP servers

Use Network Time Protocol (NTP) servers to maintain a synchronized time and date across your systems and virtual machines.

About this task

A configured NTP server that is accessible by your virtual machines is required to successfully deploy a virtual application pattern or virtual system pattern. When virtual application patterns or virtual system patterns are deployed, the NTP server is used to establish the system time for your virtual machines. Without a synchronized date and time, problems might occur resulting in incomplete deployments or failure to start virtual application instances or virtual system instances. If an NTP server is not used, the system clocks for the IBM Cloud Orchestrator environment and the hypervisors must be synchronized manually.

Procedure

1. On the system where the Workload Deployer component is installed, edit the /opt/ibm/rainmaker/purescale.app/private/expanded/ibm/scp.ui-1.0.0/config/openstack.config file.
2. In the /config/openstack section, set the NTP servers as shown in the following example:
   
   ```json
   "ntp_servers": [
       "dsnnode.customer.ibm.com",
       "127.0.0.1",
       "127.0.0.2",
       "127.0.0.3"
   ]
   ```

   Note:

   a. The list of the defined NTP servers is propagated to the provisioned virtual machines, and is used to configure NTP on the virtual machines.
   b. The list must include at least one valid NTP server for each region.
   c. A maximum of four entries are used. IP addresses or FQDNs are allowed.
   d. By default, the list contains only the NTP servers that are configured in the /etc/ntp.conf file on the system where the Workload Deployer component is installed.
   e. During IBM Cloud Orchestrator installation, only one NTP server is configured: that is, the Deployment Service node.
3. Restart the Workload Deployer service by running the following command:
   `service iwd restart`
Results

After completing these steps, you have defined an NTP server to maintain clock synchronization across your environments.

Configuring Memcached cache size

IBM Cloud Orchestrator uses Memcached with an in-memory storage and expiration dates to manage the tokens in case the performance is impacted by large numbers of tokens that are generated by multiple concurrent users, you could increase the cache size according to your requirement.

Procedure

1. Login to Central Server 2, edit the /etc/sysconfig/memcached file and set the CACHESIZE=4096 parameter. Then restart the service to take effect by running the following command:
   ```
   service memcached restart
   ```
2. Verify that the Memcached cache size is increased by running `memcached-tool 127.0.0.1:11211 stats`. If Memcached works properly, the values of total_items, total_connections, and get_hits are increasing.

Replacing the existing certificates

You can replace existing certificates.

Discover Keystore Password

The SSL certificate used by the IBM HTTP Server is contained in a file called a certificate store. This file is protected by a password. By default, this password is set to the value of the master administrator password that is set during the IBM Cloud Orchestrator installation. If you cannot remember this password, complete the following steps to retrieve this value from an encrypted Chef data bag.

1. Log on to the Deployment Server in your environment (that is, the server that was used to do the installation), and gain root privilege with the `su` command.
2. Load the OpenStack environment variables by running the following command:
   ```
   source /root/keystonerc
   ```
3. Show a list of the installation tasks that were run:
   ```
   ds job-list
   ```
4. Depending on the installation topology that you used, one or more jobs is listed:
   - If only one job is listed, make a note of the ID (that is, the long hexadecimal string in the ID column).
   - If two or more jobs are listed, make a note of the ID for the Central Server job.
   - If multiple IBM Cloud Orchestrator installations were deployed from the same deployment server, use the time stamps to identify the correct Central Server job.

Tip: If it is still not possible to identify the correct job:

a. Use the remaining steps in this section to discover the password used for the jobs. Repeat the steps as necessary to identify the password for each job ID.
b. When you generate the certificate requests, as described in the next section, try each password in turn until you find the correct password to unlock the certificate store.

5. Change to the Chef configuration directory:
   
   ```sh
cd /etc/chef
   ```

6. Locate the file containing the encryption key:
   
   ```sh
cat /etc/chef/databag_secret
   ```

7. Obtain a list of the Chef data bags:
   
   ```sh
knife data bag list
   ```

8. Identify the data bag whose name is `user-OC_ENV-<guid>`, where `<guid>` is the ID of the installation task as identified in step 4.

9. List the items in the data bag:
   
   ```sh
knife data bag show <data_bag_name>
   ```

   where the value `<data_bag_name>` is the entire string `user-OC_ENV-<guid>`.

10. Display the value of the `ihs_admin` password:
    
    ```sh
knife data bag show <data_bag_name> ihs_admin --secret-file /etc/chef/databag_secret
    ```

    This command returns the `ihs_admin` ID and password.

11. Make a note of the password, which is required for all operations on the IBM HTTP Server SSL certificate store.

**Generate Certificate Request**

1. Logon to the Central Server 2 (in high-availability environment, the primary Central Server 2) and gain root privilege with `su`.

2. Change directory to `/opt/IBM/HTTPServer/bin`

3. Backup the existing certificate store:
   
   ```sh
cp key.kdb key.kdb.bak
   ```

4. Check that the password from the first Section works and get a list of the certificates in the certificate store:
   
   ```sh
./gskcmd -cert -list -db key.kdb -pw <password>
   ```

5. The output should show two certificates. The name of the first should be the fully qualified domain name (`fqdn`) of the virtual address of Central Server 2. Make a careful note of this name because you will be required to re-enter this name in the following steps whenever you see `<fqdn>`. The second certificate name should start with a long numeric label followed by a number of parameters with the value `unknown`. This is an internal certificate used by the IBM HTTP Server to forward traffic to the IBM Cloud Orchestrator UI server on port 7443. You do not require to change this certificate but it must be present so do not modify or delete it.

6. Remove the existing SSL certificate:
   
   ```sh
./gskcmd -cert -delete -label <fqdn> -db key.kdb -pw <password>
   ```

7. Create a Certificate Request:
   
   ```sh
./gskcmd -certreq -create -label <fqdn> \
-dn "CN=<fqdn>,O=<your organisation>,OU=<your division>,C=<your country code>" \
-db key.kdb -file certreq.arm -pw <password>
   ```

8. Locate the file `certreq.arm` in the current directory and upload this file to your Certificate Authority for signing.
Install the new certificate

1. When the CA returns the signed certificate, download it to this directory as cert.arm.
2. Import Root and Intermediate Certificates. It is essential that you perform this step otherwise the HTTP Server will not work properly. Consult the Certificate Authority’s online help for details of which root and intermediate certificates are required. Download the certificates and import each of them using commands similar to:
   
   ```
   ./gskcmd -cert -import -target key.kdb -pw <password> -file <downloaded certificate file>
   ```
3. Add the New SSL Certificate:
   
   ```
   ./gskcmd -cert -receive -db key.kdb -pw <password> -file cert.arm
   ```
4. Check that the certificate has been added to the certificate store:
   
   ```
   ./gskcmd -cert -list -db key.kdb -pw <password>
   ```
5. Make the certificate the default certificate so that the IBM HTTP Server knows which one to use:
   
   ```
   ./gskcmd -cert -setdefault -db key.kdb -pw <password> -label <fqdn>
   ```
6. Examine the certificate and add a reminder to your calendar for when the certificate will expire:
   
   ```
   ./gskcmd -cert -getdefault -db key.kdb -pw <password>
   ```

Test the new certificate

Note: If your IBM Cloud Orchestrator has been installed to be highly available, shutdown the secondary Central Server 2 before proceeding otherwise it is difficult to predict which Central Server 2 is handling the requests during the following test.

1. Restart the IBM HTTP Server so it uses the new certificate:
   
   ```
   service ihs-apachectl restart
   ```
2. Test the new certificate, use a browser to access the address https://<fqdn>:8443/dashboard.
3. If the browser cannot connect, check the IBM HTTP Server log file:
   
   ```
   tail /opt/IBM/HTTPServer/logs/error_log
   ```
   
   If you see the message, SSL0208E: SSL Handshake Failed, Certificate validation error., this indicates a problem with the root and intermediate certificates. Recheck that the correct intermediate and root certificates from the CA have been imported. If you have to import more certificates, make sure that the default certificate selection does not change and correct if necessary.
4. Once the browser connects, use the browser to examine the certificate and confirm it is as expected. If it is not the correct certificate, recheck which certificate is the default as detailed before.

Update the WebSphere trust store

Next you need to update the certificate in the WebSphere trust store so that WebSphere can establish SSL connections to the IBM HTTP Server in order to make REST calls.

1. Logon to the WebSphere Console. Use the browser to access https://<fqdn>:9043/ibm/console
   
   The user ID is bpm_admin and the password is the master password specified when you installed IBM Cloud Orchestrator.
2. Replace the SSL Certificate in the WebSphere trust store:
a. Navigate to Security > SSL Certificate and key management.
b. Choose Key stores and certificates.
c. Keep the default of SSL keystores in the dropdown list.
d. Click on the link called CellDefaultTrustStore and then on Personal certificates.
e. Select the entry where the Alias matches the FQDN of the certificate you requested above and press Delete.
f. Save the change direct to the master configuration.
g. Click Import....
h. Choose Key store file and enter /opt/IBM/HTTPServer/bin/key.kdb for the Key file name.
i. Change Type to CMSKS.
j. Enter the keystore password.
k. Click Get Key File Aliases.
l. Choose the correct certificate from the dropdown list (the one which matches the FQDN).
m. Enter the same label (the FQDN) into the Imported certificate alias field.
n. Press OK.

Note: WebSphere will import the other certificates from the trust chain automatically.

o. Save the change direct to the master configuration.

3. Restart Websphere. At the terminal window enter:
   service bpm-server restart
   
**Update the certificate on the secondary Central Server 2 (if required)**

Only for Highly Available installations of IBM Cloud Orchestrator, perform these additional steps.
1. Start up the secondary Central Server 2 node, logon and gain root privilege with su.
2. Change directory to /opt/IBM/HTTPServer/bin.
3. Back up the existing certificate store:
   cp key.kdb key.kdb.bak
4. Copy the certificate store from the primary node to the secondary node:
   scp <cs2 primary node hostname>:/opt/IBM/HTTPServer/bin/key.kdb
5. If the secondary Central Server 2 was only suspended during the work on primary Central Server 2, you must restart BPM on the secondary node so that it uses the new certificate:
   service bpm-server restart
6. You do not need to restart the IBM HTTP Server on the secondary node since it will not normally be running and will be started automatically if the primary Central Server 2 node fails.

**Update the certificate on the Workload Deployer server**

To replace certificate on the Workload Deployer server, perform these steps:
1. Log on to Central Server 3 (Workload Deployer server).
2. Back up the Workload Deployer certificate located in the following path:
   `/opt/ibm/rainmaker/purescale.app/private/expanded/ibm/rainmaker.rest-4.1.0.0/config/rm.p12`

3. Back up the Workload Deployer kernel services truststore located in
   `/opt/ibm/maestro/maestro/usr/resources/security/KSTrustStore.jks`.

4. Copy the IBM HTTP Server certificate to the following path:
   `/opt/ibm/rainmaker/purescale.app/private/expanded/ibm/rainmaker.rest-4.1.0.0/config/rm.p12`

5. Import the certificate to kernel services truststore:
   ```
   source /etc/profile.d/jdk_iwd.sh
   keytool -v -importkeystore
   -srckeystore /opt/ibm/rainmaker/purescale.app/private/expanded/ibm/rainmaker.rest-4.1.0.0/config/rm.p12
   -srcstoretype PKCS12 -destkeystore /opt/ibm/maestro/maestro/usr/resources/security/KSTrustStore.jks
   -deststoretype JKS -deststorepass pureScale -srcstorepass <password>
   ```

To replace the certificate for the Workload Deployer command line, perform the following steps:

1. Log on to Central Server 3 (Workload Deployer server).
2. Switch to `/opt/ibm/rainmaker/purescale.app/private/expanded/ibm/rainmaker.ui-4.1.0.0/public/downloads`.
3. Unzip the file `deployer.cli-5.0.0.0-<buildid>.zip`, to the current folder, for example:
   ```
   unzip -q deployer.cli-5.0.0.0-20140815194859.zip
   ```
   **Note:** There will be only one file that matches this naming scheme.
4. Move the file `deployer.cli-5.0.0.0-<buildid>.zip` to a backup folder.
5. Copy the IBM HTTP Server certificate to the files `deployer.cli/lib/cb.p12` and `deployer.cli/lib/deployer-ssl.p12`, overwriting the existing files. Make sure the files have the correct access rights (640):
   ```
   chmod 640 deployer.cli/lib/cb.p12 deployer.cli/lib/deployer-ssl.p12
   ```
6. Import certificate to the Workload Deployer CLI keystore:
   ```
   source /etc/profile.d/jdk_iwd.sh;keytool -v -importkeystore
   -srckeystore deployer.cli/lib/cb.p12
   -srcstoretype PKCS12 -destkeystore deployer.cli/lib/deployer-ssl.jks
   -deststoretype JKS -deststorepass rainmaker -srcstorepass <password>
   ```
7. Delete file `deployer.cli-5.0.0.0-<buildid>.zip`.
8. Archive the content of the `deployer.cli` directory into a zip file having the same name as original file:
   ```
   zip -q -r deployer.cli-5.0.0.0-<buildid>.zip deployer.cli
   ```

**Maintaining a deployed environment**

You can use deployment services to maintain a deployed environment in the whole life cycle.

**Scaling out a deployed environment**

More nodes can be added as compute nodes after the first deployment.

Run one of the following procedures on the Deployment Server:

- **Use the deployment service wizard to add new nodes to a deployed environment.** Perform the following steps:

  1. Run the following command to launch the interactive CLI:
     ```
     source /root/keystonerc; ds wizard
     ```
     **Note:** Terminal window must be at least 35 lines and 130 columns.
2. Choose the following option to modify an existing IBM Cloud Orchestrator deployment:

- Modify an IBM Cloud Orchestrator deployment, add region server or KVM compute node.

and follow the interactive procedure by entering the required information. For information about the deployment parameters, see “Deployment parameters” on page 43.

- **Use the deployment service CLI to add new nodes to a deployed environment.**

Perform the following steps:

1. Check that the following jobs exist:

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>80a8b343-8c92-4558-a18d-3b5d0836403c</td>
<td>centralserver</td>
<td>FINISHED</td>
</tr>
<tr>
<td>d6693ee-2aa6-4182-b897-ac6a633b1959</td>
<td>regionserver</td>
<td>FINISHED</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>pjob_id</th>
<th>created_at</th>
<th>updated_at</th>
</tr>
</thead>
</table>
```

2. Retrieve the available resource for a job. Get a list of the available resources for a job with the following command:

```
source /root/keystonerc; ds job-resources-list <job_id>
```

For example:

```
ds job-resources-list d6693ee-2aa6-4182-b897-ac6a633b1959
```

```
<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>run_list</th>
<th>run_order</th>
</tr>
</thead>
<tbody>
<tr>
<td>kvm_region_server</td>
<td>Existing Machine</td>
<td>role[allinone]</td>
<td>1</td>
</tr>
<tr>
<td>compute</td>
<td>Existing Machine</td>
<td>role[os-compute-worker-sco]</td>
<td>2</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>node_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>921b8f28-fdad-4856-9299-4cb0b596abe7</td>
</tr>
<tr>
<td>04e73c27-1b77-4f00-9444-86f74ace54f8</td>
</tr>
</tbody>
</table>
```

3. Register more nodes. More nodes can be added as additional compute nodes.

For example:

```
ds node-create -t IBM::SCO::Node
    -p '{Address: 172.16.2.108, Port: 22, User: root, Password: passw0rd}' computeB
```

```
ds node-list
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>type</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>df99db9b-fae0-46cf-a0ff-dacc9f3c294f</td>
<td>central1</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
<tr>
<td>d0b8b1c7-6ca8-4d5b-a911-0e70b020c1c9</td>
<td>computeB</td>
<td>IBM::SCO::Node</td>
<td>FREE</td>
</tr>
<tr>
<td>e31532bc-941d-4f1c-bf57-3aaa7cc388252</td>
<td>central2</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
<tr>
<td>921b8f28-fdad-4856-9299-4cb0b596abe7</td>
<td>region</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
<tr>
<td>04e73c27-1b77-4f00-9444-86f74ace54f8</td>
<td>compute</td>
<td>IBM::SCO::Node</td>
<td>INUSE</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>job_id</th>
<th>resource</th>
<th>created_at</th>
</tr>
</thead>
<tbody>
<tr>
<td>80a8b343-8c92-4558-a18d-3b5d0836403c</td>
<td>central_server_1</td>
<td>2014-04-16T03:03:48.641240</td>
</tr>
</tbody>
</table>
```
4. Associate the nodes to the existing job by running the following command:

```
source /root/keystonerc; ds job-associate-node [-N <RES1=NODE1;RES2=NODE2...>] <NAME or ID>
```

For example:
```
source /root/keystonerc;
source /root/keystonerc;
ds job-associate-node -N compute=d68bb1c7-6ca8-4d5b-a491-1e078062c1c9
d66993ee-2aa6-4182-8897-ac6a633b1959
```

If you need to associate one or more nodes to a new job, you can disassociate the node (or multiple nodes) from the current job by running the following command:

```
source /root/keystonerc; ds job-disassociate-node [-N <NODE1,NODE2...>]
```

Optional arguments:
```
-N <NODE1,NODE2...>, --nodes <NODE1,NODE2...>
```

Removes node mapping of the deployment job. This can be specified multiple times, or once with nodes separated by commas.

For example:
```
ds job-disassociate-node -N 4d591adf-c01a-42e7-8162-23cbf8ec5637,faf02217-4cc6-421e-9def-0bf96b29b2b0
```

5. Update the job by running the following command:

```
source /root/keystonerc; job-update <NAME or ID>
```

After the execution finishes, the job is marked as UPDATED. For example:
```
ds job-list
```

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>80a8b343-8c92-4558-a18d-3b5d0836403c</td>
<td>central_server_2</td>
<td>FINISHED</td>
</tr>
<tr>
<td>d66993ee-2aa6-4182-8897-ac6a633b1959</td>
<td>kvm_region_server</td>
<td>UPDATED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pjob_id</th>
<th>created_at</th>
<th>updated_at</th>
</tr>
</thead>
</table>

Chapter 2. Installing 101
**Backing up and restoring Deployment Service**

The `ds-backup` and `ds-restore` commands are provided in the Deployment Server to backup and restore the data for Deployment Service. The `ds-backup` command backs up the important data included in the database, chef data and other important files.

To back up with the `ds-backup` command, run:

```
ds-backup <DBINSTANCENAME> <OUTPUT_LOCATION> [MODE]
```

where:

- `<DBINSTANCENAME>`
  - Defines the instance name of DB2, such as `db2inst1`.

- `<OUTPUT_LOCATION>`
  - Defines the backup package location, the `ds-backup` command will package all files into a `tag.gz` file and put it into this location.

- `[MODE]`
  - Defines the database backup mode, can be `online` or `offline`. This argument can be ignored and the default value is `online` if ignored.

**Note:** It is required to stop all of the services connected to DB2 before doing offline backup.

**Note:** There are some setup steps needed to enable the online database backup. For more information, see the DB2 documentation.

To restore with the `ds-restore` command:

```
ds-restore <DBINSTANCENAME> <BACKUP_PACKAGE>
```

where:

- `<DBINSTANCENAME>`
  - Defines the instance name of the DB2, such as `db2inst1`.

- `<BACKUP_PACKAGE>`
  - Defines the backup output. For example `/tmp/backup/
ds20140528140442.tar.gz`

**Note:** It is required to stop all of the services connected to DB2 before doing a restore.

**Move the database to an external system**

If the current DB2 installation is not able to meet with adequate performance the increasing workload, you can setup a powerful machine where to move the database.

**Before you begin**

Be sure that you specified a service IP address when you registered the DB2 node. For more information, see “Registering the nodes” on page 54.

**Procedure**

1. Shut down IBM Cloud Orchestrator.
2. Back up the database on the DB2 node.
3. Remove the service IP address from the DB2 node.
4. Configure the service IP address to the new machine.
5. Setup DB2 on a new machine.
6. Restore the database back to new DB2.
7. Start the new DB2 server.
8. Start IBM Cloud Orchestrator.

**Content pack installation**

To make the content pack processes available in IBM Cloud Orchestrator, import the toolkit provided with the content pack in Business Process Manager.

Follow this procedure:
1. In the Business Process Manager Designer, click **Process Center**.
2. In Toolkits, click **Import Toolkit**.
3. In Import Toolkit, click **Browse** to select the file to import.

   **Note:** The Toolkit file extension is in twx extension format.
4. Click **OK**.

When the Toolkit is imported in Business Process Manager, the processes of the toolkit are available in IBM Cloud Orchestrator, which can be configured as self-service offering or **Orchestration Action** from the IBM Cloud Orchestrator user interface.

**Log in to the Self-service user interface:**
1. Click **CONFIGURATION** > **Categories**.
2. Assign a name, a description, and select a graphical icon for the category.
3. Click **CONFIGURATION** > **Offerings**.
4. Provide the offering name and select the newly created category and icon.
5. Select a process to create self-service offerings.

**Configuring Central Nodes to connect to deployed virtual machines**

When you apply VlanManager as your network configuration and the deployed virtual machines are all in some virtual LANs, then the Central Nodes must be configured to manually connect to these deployed virtual machines.

**Procedure**

1. **Check whether your network is VlanManager:**

   ```
   [root@sco24-b6-node2 ~]# nova network-list
   ID          | Label     | Cidr            |
   +------------+-----------+-----------------+
   e45ffdd5-0f97-482b-80ff-16058a980f8a | VM Network | 172.17.48.0/22 |
   
   [root@sco24-b6-node2 ~]# nova network-show e45ffdd5-0f97-482b-80ff-16058a980f8a
   +---------------------+--------------------------------------+
   | Property             | Value                               |
   +---------------------+--------------------------------------+
   | bridge               | br4096                               |
   | bridge_interface     | eth1                                 |
   | broadcast            | 172.17.51.255                        |
   | cidr                 | 172.17.48.0/22                       |
   | cidr_v6              | -                                    |
   ```
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>created_at</td>
<td>2014-07-16T02:32:59.568911</td>
</tr>
<tr>
<td>deleted</td>
<td>0</td>
</tr>
<tr>
<td>deleted_at</td>
<td>-</td>
</tr>
<tr>
<td>dhcp_start</td>
<td>172.17.48.2</td>
</tr>
<tr>
<td>dns1</td>
<td>8.8.4.4</td>
</tr>
<tr>
<td>dns2</td>
<td>-</td>
</tr>
<tr>
<td>gateway</td>
<td>172.17.48.1</td>
</tr>
<tr>
<td>gateway_v6</td>
<td>-</td>
</tr>
<tr>
<td>host</td>
<td>-</td>
</tr>
<tr>
<td>id</td>
<td>e45ffdd5-0f97-482b-80ff-16058a980f8a</td>
</tr>
<tr>
<td>injected</td>
<td>True</td>
</tr>
<tr>
<td>label</td>
<td>VM Network</td>
</tr>
<tr>
<td>multi_host</td>
<td>False</td>
</tr>
<tr>
<td>netmask</td>
<td>255.255.252.0</td>
</tr>
<tr>
<td>netmask_v6</td>
<td>-</td>
</tr>
<tr>
<td>priority</td>
<td>-</td>
</tr>
<tr>
<td>project_id</td>
<td>-</td>
</tr>
<tr>
<td>rxtx_base</td>
<td>-</td>
</tr>
<tr>
<td>updated_at</td>
<td>2014-07-16T02:35:00.931734</td>
</tr>
<tr>
<td>vlan</td>
<td>100</td>
</tr>
<tr>
<td>vpn_private_address</td>
<td>-</td>
</tr>
<tr>
<td>vpn_public_address</td>
<td>-</td>
</tr>
<tr>
<td>vpn_public_port</td>
<td>-</td>
</tr>
</tbody>
</table>

As above if its property "vlan" has a value then this network is in the type of vlan.

2. **Configure your Central nodes networks:**

Take network in step 1 as an example. Virtual machines deployed to this network have IP addresses in range 172.17.48.0/22 and in the VLAN 100. Then you should set your Central Nodes networks to connect to these IP addresses in the VLAN 100.

a. **Add a vlan device in your Central Nodes:**

```
[root@sco24-b6-node2 ~]# vconfig add eth1 100
Added VLAN with VID == 100 to IF -:eth1:-
[root@sco24-b6-node2 ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:50:56:93:4c:8e brd ff:ff:ff:ff:ff:ff
8: eth1.100@eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
    link/ether 00:50:56:93:4c:8e brd ff:ff:ff:ff:ff:ff
```

As above, device 8 is the created device.

b. **Enable your created device:**

```
[root@sco24-b6-node2 ~]# ifconfig eth1.100 up
[root@sco24-b6-node2 ~]# ifconfig eth1 Link encap:Ethernet HWaddr 00:50:56:93:4c:8e
    inet6 addr: fe80::250:56ff:fe93:4c8e/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:14879888 errors:0 dropped:0 overruns:0 frame:0
    TX packets:7014089 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:2803590536 (26.1 GiB) TX bytes:9284891183 (8.6 GiB)

eth1.100 Link encap:Ethernet HWaddr 00:50:56:93:4c:8e
    inet6 addr: fe80::250:56ff:fe93:4c8e/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:0 errors:0 dropped:0 overruns:0 frame:0
    TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:0 (0.0 b) TX bytes:468 (468.0 b)
```
lo   Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:67398644  errors:0  dropped:0  overruns:0  frame:0
TX packets:67398644  errors:0  dropped:0  overruns:0  carrier:0
 collisions:0  txqueuelen:0
 RX bytes:42055289407 (39.1 GiB)  TX bytes:42055289407 (39.1 GiB)

As above create a virtual bridge and enslave the created device to it.

Open VNC port on ESXi host to enable the VNC console access

You can enable the VNC console access.

On the vSphere Client for all the hosts in the cluster, if you want to access the virtual machine via VNC console through the Administration UI or the command line, run: `nova get-vnc-console XXXXXX xvpnc`

Select the host and then Configuration -> Security Profile -> Firewall -> Edit -> gdbserver. Select the check box and click OK.

For production, limit the ports to 5900-6000 for security considerations. The port range depends on how many virtual machines can run on this host. For more information about the configuration, see the VNC section under Prerequisites and limitations in [http://docs.openstack.org/icehouse/config-reference/content/vmware.html](http://docs.openstack.org/icehouse/config-reference/content/vmware.html)

Managing flavors

You can create a flavor by running a command.

You can use the following command to create a flavor:

```
nova flavor-create <flavor_name> <flavor_id> <ram_size> <disk_size> <vcpus>
```

For information about the arguments, run `nova help flavor-create`.

**Note:** The `disk_size` must be larger than size of all the disks when you are trying to boot an instance with an existing template. If the `disk_size` is 0, the instance is booted without a size check.

Additional flavors can also be added for KVM, VMWare and zVM by one of these mechanisms:
• via IBM Cloud Orchestrator Horizon Administration user interface
• nova command line from the OpenStack Nova compute node on the appropriate
region server. For more information, go to Creating new flavors in OpenStack.

Note: For z/VM, disk size is 5 GB. Configure flavors for supporting disk size only.
To make a disk size larger than 5 GB, follow the steps documented in OpenStack
Enablement for z/VM.
To manage flavours for Softlayer, Amazon EC2 and non-IBM supplied OpenStack,
go to Configuring flavors.

Note: Each Region defines its own distinct set of flavors.

Creating new flavors in OpenStack

When you are deploying a virtual system pattern in OpenStack, you can create
new flavors using the nova flavor-create command.

With some images there are issues with resizing images larger than the size of the
image. If you do not have a reason to change the size, you can use a flavor with a
disk size of 0 (zero). You can create new flavors in OpenStack using the nova
flavor-create command.

```
nova flavor-create [--ephemeral <ephemeral>] [--swap <swap>] 
  [--rxtx-factor <factor>] [--is-public <is-public>] 
  <name> <id> <ram> <disk> <vcpus>
```

where:
• --ephemeral <ephemeral> (optional) is the ephemeral space size in GB (the
default is 0).
• --swap <swap> (optional) is the swap size in MB (the default is 0).
• --rxtx-factor <factor> is the RX/TX factor (the default is 1).
• --is-public <is-public> makes the flavor accessible to the public (the default is
ture).
• <name> is the name of the new flavor.
• <id> is the unique integer ID for the new flavor.
• <ram> is the memory size in MB.
• <disk> is the disk size in GB.
• <vcpus> is the number of vCPUs.

Flavors must be formed in a way that memory, CPU, and disk in flavor should be
larger or equal to the image requirement (in the disk case, it might be 0). These
image requirements can be seen in the IBM Cloud Orchestrator UI under the
Hardware section for the registered/imported virtual image.

Example of the command:

```
nova flavor-create 4gb1cpu 12345 4096 0 1
```

Note: The ID (12345) is any number that has not been ever used before.

To view the currently defined flavors, use the nova flavor-list command.
Customizing flavors for Power features

Use PowerVC flavors to manage PowerVM® LPAR settings.

All flavors used with PowerVC must be created with the PowerVC user interface. This is due to extra specifications PowerVC requires from its flavors.

Any flavor created on the PowerVC user interface will be automatically imported to IBM Cloud Orchestrator with a prefix assigned in `/etc/powervc/powervc.conf`. Only use flavors with this prefix when deploying PowerVC Images from IBM Cloud Orchestrator.

Managing Nova networks

Manage a Nova network in your IBM Cloud Orchestrator environment.

Adding Nova networks

You can add additional networks in addition to the network created during the IBM Cloud Orchestrator installation procedure.

Before you begin

- Ensure your environment meets all the requirements described in “Planning networks” on page 24.
- Identify all the IP addresses that are already in use in the IP range that you plan to use. You must mark these IP addresses as reserved in OpenStack so that IP conflicts do not occur at deployment time. To reserve IP addresses, use the `nova fixed-ip-reserve <fixed_IP>` command.
- All the Compute Nodes must use the same interface to connect to the Region Server. For example, eth0, eth1, or eth2.

Restriction: If using nova networks and distributed switches on VMware, the port groups must exist in VMware and at network creation time you must specify its exact name.

Procedure

1. To create a network, run the following command on one line on the Region Server:
   ```bash
   nova-manage network create --label='LABEL' --fixed_range_v4=X.X.X.X/YY --num_networks=1 --network_size=256 --gateway=GATEWAY_IP --vlan=VLAN --bridge_interface=NIC --dns1 X.X.X.X --dns2 X.X.X.X --project PROJECT_ID
   ```

   For example:
   ```bash
   nova-manage network create --label='VLAN106' --fixed_range_v4=10.10.6.0/24 --num_networks=1 --network_size=256 --gateway=10.10.6.1 --vlan=106 --bridge_interface=eth3 --dns1 9.110.51.41 --dns2 9.110.51.41 --project 9d9d88a46e5b4022ae164f6b2ed42469
   ```

   The `bridge_interface` parameter is the interface that all compute nodes configured to the VLAN in their switch. After the VLAN network is created, no new VLAN or bridge device is created in the compute node until a virtual machine is deployed in the compute node.

   The `project` parameter associates the specified `PROJECT_ID` with the network, so that only the users assigned to that project can access the network. You can associate only one project with a network.
2. Optional: If you want users from multiple projects to access the same network, you must disassociate all projects from the network, which sets the `project_id` value for the network to `None`. To disassociate all projects from the network, run the following command on the Region Server:

   `nova-manage network modify X.X.X.X/YY --disassociate-project`

   For example:
   `nova-manage network modify 10.10.6.0/24 --disassociate-project`

   The network can now be accessed by all users in all projects.

3. Configure the nodes to use the new gateway. Otherwise, they might use their host as a gateway. Run the following commands on the Region Server and on each compute node:

   `echo "dhcp-option=tag:LABEL,option:router,GATEWAY_IP"` >>
   `/etc/dnsmasq.d/nova.gateway`

   `echo "domain=DNS_SUFFIX,X.X.X.X/YY"` >>
   `/etc/dnsmasq.d/nova.domain`

4. Stop all `dnsmasq` services and restart the `openstack-nova-network` service on the Region Server or compute node:

   `killall dnsmasq`
   `service openstack-nova-network restart`

**What to do next**

Associate the network to a project. For more information, see "[Associating a Nova network to a project](#)"

**Associating a Nova network to a project**

To achieve network segregation, it is important that you associate each network to a specific project ID. This is mandatory for KVM regions.

**About this task**

In KVM regions, VLANManager is the network manager for VLAN-based networks and for flat networks. VLANManager is designed to deploy the virtual-machine instances of different projects in different subnets. Therefore, in a KVM region, flat networks and VLAN networks must be associated with a project before you use the network.

In VMware regions you can use either FlatManager or VLANManager as the network manager. FlatManager uses one flat IP address pool that is defined throughout the cluster. This address space is shared among all user instances, regardless to which project the instance belong. Each project can take any address that is available in the pool. Therefore, when using FlatManager, you do not need to associate the network with a project before you use the network.

In VMware regions, FlatManager is the default for environments upgraded from SmartCloud Orchestrator 2.3, VLANManager is the default for IBM Cloud Orchestrator environments that do not use Neutron.

When you associate a network with a project, only the users assigned to that project can access the network. You can associate a network with only one project.

If you want users from multiple projects to access the same network, you must disassociate the network from all projects, which sets the `project_id` value for the network to `None`. The network can then be accessed by all users in all projects.
Procedure

1. To view the details of an OpenStack network, run the `nova network-show` command on the Region Server as the `admin` user, as shown in the following example:

   ```
   nova network-show e325a701-ab07-4fb9-a7df-621e0eb31c9b
   ```

   +---------------------+--------------------------------------+
   | Property         | Value                  |
   +---------------------+--------------------------------------+
   | bridge             | br4090                  |
   | bridge_interface  | eth1                    |
   | broadcast          | 10.10.255.255           |
   | cidr               | 10.10.0.0/16            |
   | cidr_v6            | None                    |
   | created_at         | 2013-04-20T09:43:40.000000 |
   | deleted            | False                   |
   | deleted_at         | None                    |
   | dhcp_start         | 10.10.0.56              |
   | dns1               | 10.10.0.57              |
   | dns2               | 10.10.0.1               |
   | gateway            | 10.10.0.1               |
   | gateway_v6         | None                    |
   | host               | None                    |
   | id                 | e325a701-ab07-4fb9-a7df-621e0eb31c9b |
   | injected           | False                   |
   | label              | public                  |
   | multi_host         | True                    |
   | netmask            | 255.255.0.0             |
   | netmask_v6         | None                    |
   | project_id         | 9d9d88a46e5b4022aef64f6b2ed42469 |
   | priority           | None                    |
   | rxtx_base          | None                    |
   | updated_at         | 2013-04-20T09:44:41.000000 |
   | vlan               | 4090                    |
   | vpn_private_address| 10.10.0.2               |
   | vpn_public_address | 127.0.0.1               |
   | vpn_public_port    | 1000                    |
   +---------------------+--------------------------------------+

   The `project_id` value must be set to the ID of the project to which the users are assigned, or set to `None` to grant access to this network to all users in all projects.

2. To associate a project with a network, run the following command on one line on the Region Server:

   ```
   nova-manage network modify X.X.X.X/YY --project $PROJECT_ID
   ```

   where `$PROJECT_ID` is the project ID of the project that you want to associate with the network.

   For example:

   ```
   nova-manage network modify 10.10.0.0/16 --project 9d9d88a46e5b4022aef64f6b2ed42469
   ```

3. If you want the network to be accessed by users from several projects, disassociate all projects from the network by running the following command on the Region Server:

   ```
   nova-manage network modify X.X.X.X/YY --disassociate-project
   ```

   For example:

   ```
   nova-manage network modify 10.10.6.0/24 --disassociate-project
   ```

   If you want the network to be used by users from several projects, run the following command to disassociate the project from the network:
nova-manage network modify 10.10.0.0/16 --disassociate-project

To verify that the network is disassociated from the project, run the following command:

```
nova network-show e325a701-ab07-4fb9-a7df-621e0eb31c9b
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bridge</td>
<td>br4090</td>
</tr>
<tr>
<td>bridge_interface</td>
<td>eth1</td>
</tr>
<tr>
<td>broadcast</td>
<td>10.10.255.255</td>
</tr>
<tr>
<td>cidr</td>
<td>10.10.0.0/16</td>
</tr>
<tr>
<td>cidr_v6</td>
<td>None</td>
</tr>
<tr>
<td>created_at</td>
<td>2013-04-20T09:43:40.000000</td>
</tr>
<tr>
<td>deleted</td>
<td>False</td>
</tr>
<tr>
<td>deleted_at</td>
<td>None</td>
</tr>
<tr>
<td>dhcp_start</td>
<td>10.10.0.56</td>
</tr>
<tr>
<td>dns1</td>
<td>10.10.0.57</td>
</tr>
<tr>
<td>dns2</td>
<td>10.10.0.1</td>
</tr>
<tr>
<td>gateway</td>
<td>10.10.0.1</td>
</tr>
<tr>
<td>gateway_v6</td>
<td>None</td>
</tr>
<tr>
<td>host</td>
<td>None</td>
</tr>
<tr>
<td>id</td>
<td>e325a701-ab07-4fb9-a7df-621e0eb31c9b</td>
</tr>
<tr>
<td>injected</td>
<td>False</td>
</tr>
<tr>
<td>label</td>
<td>public</td>
</tr>
<tr>
<td>multi_host</td>
<td>True</td>
</tr>
<tr>
<td>netmask</td>
<td>255.255.0.0</td>
</tr>
<tr>
<td>netmask_v6</td>
<td>None</td>
</tr>
<tr>
<td>priority</td>
<td>None</td>
</tr>
<tr>
<td>project_id</td>
<td>None</td>
</tr>
<tr>
<td>rxtx_base</td>
<td>None</td>
</tr>
<tr>
<td>updated_at</td>
<td>2013-04-20T15:09:35.000000</td>
</tr>
<tr>
<td>vlan</td>
<td>4090</td>
</tr>
<tr>
<td>vpn_private_address</td>
<td>10.10.0.2</td>
</tr>
<tr>
<td>vpn_public_address</td>
<td>127.0.0.1</td>
</tr>
<tr>
<td>vpn_public_port</td>
<td>1000</td>
</tr>
</tbody>
</table>

The `project_id` value is set to `None`, which means that the network is shared and can be accessed by all users in all projects.

**Managing Neutron networks**

Manage a Neutron network in your IBM Cloud Orchestrator environment.

**Adding Neutron networks**

You can add a Neutron network to your IBM Cloud Orchestrator environment.

To define a Neutron network, log in to the Administration user interface and click `Admin > System Panel > Networks`.

You can assign the network to a specific project and you can select one of the following provider network types:

- **FLAT** All instances reside on the same network, which can also be shared with the hosts. No VLAN tagging or other network segregation takes place. It does not support overlapping IP addresses.
- **Local** Instances reside on the local compute host and are effectively isolated from any external networks.
- **VLAN** Networking allows users to create multiple provider or tenant networks
using VLAN IDs (802.1Q tagged) that correspond to VLANs present in the physical network. This allows instances to communicate with each other across the environment. They can also communicate with dedicated servers, firewalls, load balancers and other networking infrastructure on the same layer 2 VLAN. It does not support overlapping IP addresses.

**VXLAN**

VXLAN use network overlays to support private communication between instances. A Networking router is required to enable traffic to traverse outside of the VXLAN tenant network. A router is also required to connect directly-connected tenant networks with external networks, including the Internet. The router provides the ability to connect to instances directly from an external network using floating IP addresses.

You must use OpenStack command line to create VXLAN networks as described in the following topics. For information about the Neutron scenarios, see “Neutron scenarios” on page 27.

**Scenario 1: Use network server as gateway:**

This topic describes how you use a network server as a gateway.

**About this task**

The following procedure refers to the command line commands. You can perform the same action from the Administration user interface.

**Procedure**

1. **Create a network to access public network (internet):**

   Check the value of physical_interface_mappings in the configuration file -- /etc/neutron/plugins/linuxbridge/linuxbridge_conf.ini on the network server; it may be one of the following configurations depending on the deployment topology:

   **Note:** All neutron commands must be run on the Region node.

   a. `physical_interface_mappings = physnet1:eth0`:
      This means that the management_interface, data_interface, external_interface all use the same NIC -- eth0, so the network creation command is:
      ```
      neutron net-create <public_net_name> --router:external=True
      --provider:network_type flat --provider:physical_network physnet1
      ```

   b. `physical_interface_mappings = physnet1:eth0,physnet2:eth1`:
      This means that management_interface and external_interface use the same NIC -- eth0, and data_interface uses eth1, so the network creation command is:
      ```
      neutron net-create <public_net_name> --router:external=True
      --provider:network_type flat --provider:physical_network physnet1
      ```

   c. `physical_interface_mappings = physnet1:eth0,physnet2:eth1,physnet3:eth2`:
      This means that management_interface uses eth0, data_interface uses eth1 and external_interface uses eth2, so the network creation command is:
      ```
      neutron net-create <public_net_name> --router:external=True
      --provider:network_type flat --provider:physical_network physnet3
      ```
Note: If network you are going to access the public network (internet) is using a VLAN, for example 1000, then the command becomes:

```
neutron net-create public --router:external=True --provider:network_type vlan
   --provider:physical_network physnetX --provider:segmentation_id 1000
```

physnetX is what external_interface maps to, and it is physnet1 in most cases.

After the layer 2 network is created, one or more subnetworks can be added into it with the following command:

```
neutron subnet-create --gateway 9.110.51.1 public 9.110.51.0/24
   --allocation-pool start=9.110.51.100,end=9.110.51.200 --name sub-pub
```

Note: This also means the floating IP pool.

2. Create networks for virtual machines:

For KVM, it can be either VLAN or VXLAN, while for VMware only VLAN is supported. If using VLAN, you must ensure the trunks in switch are configured properly:

```
neutron net-create <vlan_net_name> --tenant-id <tenant_id> --provider:network_type vlan
   --provider:physical_network <physnetX> --provider:segmentation_id <number>
```

or:

```
neutron net-create <vxlan_net_name> --tenant-id <tenant_id> --provider:network_type vxlan
   --provider:segmentation_id <number>
```

For VMware, since a VMware native driver does not support to create the port group in vCenter automatically, you must create the corresponding port group in vCenter for each created network by manually before boot the virtual machine, the name of port group must be the same as the name (label) of the neutron networks.

Note: For VLAN, the trunked port must be configured properly for the <number> and <physnetX> must be data_interface. For VXLAN, <number> must be from 1000 to 160000.

Add subnetworks into it which means reserve IP ranges for the virtual machines, for example:

```
neutron subnet-create <vlan_net_name | vxlan_net_name>10.10.10.0/24
```

3. Create routers for virtual machines:

Create a router:

```
neutron router-create router1
```

Make the router able to access public network:

```
neutron router-gateway-set router1 <public network uuid>
```

Note: <public network uuid> is the one created in 1 on page 111

Make virtual machines network connect to the router:

```
neutron router-interface-add router1 <subnet uuid>
```

Note: <subnet uuid> is the subnet in 2 created by subnet-create command.
Scenario 2: Use a physical gateway:

This topic describes how to use a physical gateway.

About this task

The following procedure refers to the command line commands. You can perform the same action from the Administration user interface.

Procedure

Create networks for virtual machines:
Can be FLAT or VLAN for either KVM or VMware, cannot be VXLAN:
neutron net-create <vlan_net_name> --tenant-id <tenant_id> --provider:network_type vlan
   --provider:physical_network <physnetX> --provider:segmentation_id <number>

or:
neutron net-create <flat_net_name> --tenant-id <tenant_id> --provider:network_type flat

For VLAN, the trunked port must be configured properly for the <number> and <physnetX> must be data_interface:
neutron subnet-create <vlan_net_name | vxlan_net_name> 10.10.10.0/24
   --gateway <gateway_IP_on_router>

Restriction: For either KVM or VMware, only Flat and VLAN mode can be used. VXLAN, overlapping IP and floating IP are not supported for this scenario.

Scenario 3: Hybrid SDN and physical routing:

This topic describes a hybrid configuration for scenario 1 and 2.

It means you may have different types of networks together. Take the following scenario as an example:
• Network 1: VXLAN, it must use network server as gateway (configure with “Scenario 1: Use network server as gateway” on page 111).
• Network 2: VLAN, you want that virtual machines on it can be accessed directly through physical routing so this network uses physical gateway (configure with “Scenario 2: Use a physical gateway”).
• Network 3 and network 4: one is VLAN and another is VXLAN, but they use the same network CIDR (meaning overlapping IP range), to support this, you must use the network server as gateway (configure with “Scenario 1: Use network server as gateway” on page 111).
• Network 5: you want to use floating IP to access the virtual machines on it, then you must use the network server as gateway (configure with “Scenario 1: Use network server as gateway” on page 111).

Note: If \texttt{mgmt\_interface = data\_interface = external\_interface}, see “Customizing deployment parameters” on page 41 for information about these interfaces. If you are going to have both Flat and VXLAN networks, you must ensure that:
1. Flat networks must be created first.
2. Before any virtual machines boot on it, you must first manually create a bridge (same as the network server) on each Compute Node.
Associating a Neutron network to a project
To achieve network segregation, it is important that you associate each network to a specific project ID by using the Administration user interface.

You can assign a network to a project at network creation time. You cannot reassign a network to a different project after network creation.

Creating multiple networks assigned to different projects
Create Neutron networks that are mapped to different VLANs, and assign the networks to different projects.

Before you begin
Create Project A and Project B in the same domain, as described in “Creating a project” on page 222. Create User A assigned to Project A, and User B assigned to Project B, as described in “Creating a user” on page 229. Assign the admin role to the Cloud Administrator. Assign the member role to User A and User B.

About this task
You create two networks and assign them to Project A: Network A1 uses VLAN 101, and Network A2 uses VLAN 102. Similarly, you create two other networks and assign them to Project B: Network B1 uses VLAN 201, and Network B1 uses VLAN 202.

You then log in as User A in Project A, and deploy virtual machines to connect to Network A1 and Network A2. Similarly, you log in as User B in Project B, and deploy virtual machines to connect to Network B1 and Network B2.

Procedure
1. Log in to the Administration user interface as a Cloud Administrator.
2. In the left navigation pane, click ADMIN > System Panel > Networks. The Networks page is displayed.
3. Create Network A1:
   a. Click Create Network. The Create Network window opens.

       Note: The Create Network action in the Administration user interface is limited to Neutron networks only. To create a Nova network, use the Self-service user interface.

       Note: For VMWare, a port group of the same name must exist in vCenter before deploying any instance.
   c. Select a project from the list to assign the network to that project. Example: Project A

       Restriction: You cannot reassign a network to a different project after network creation.
   d. Select the network type from the list. Example: VLAN
   e. Specify the other network details.

       Name For example testNetwork. If it is a VMWare region, there must be a port group with the same name.
Physical Network
For example physnet1. For information about physnetX, see "Scenario 1: Use network server as gateway" on page 111.

Segmentation ID
For example 101. Must be a valid VLAN ID and must have been configured on the switch correctly.

f. Click Create Network.
The Create Network window closes. The network is created, and the name is displayed on the Networks page.

4. Create a subnet for Network A1:
a. Click the network name to display the Network Detail page.
b. Click Create Subnet. The Create Subnet window opens.
c. In the Subnet tab, specify the subnet name and other details:
   Subnet Name
   Any preferred name, for example testSubnet.
   Network Address
   A planned network CIDR such as 10.10.10.0/24.
   Gateway IP
   The gateway for this subnet. You can also input the allocation pool if necessary.

   Click Next.
d. In the Subnet Details tab, specify more subnet attributes.
e. Click Create.
   The Create Subnet window closes. The subnet is created.

5. Repeat step 3 and step 4 as necessary to create Network A2, Network B1, Network B2, and the associated subnets.

6. Log in to the Self-service user interface as User A.
7. Deploy a virtual machine to connect to Network A1:
a. Click SELF-SERVICE CATALOG.
b. Click Manage Virtual Machines > Deploy Single Virtual Machine.
c. Specify that the server connects to Network A1.
d. Click Deploy.
   The virtual machine is deployed and connects to Network A1.

8. Repeat step 7 to deploy a virtual machine that connects to Network A2.
9. Log in to the Self-service user interface as User B.
10. Repeat step 7 as necessary to deploy virtual machines that connect to Network B1 and Network B2.

Results
IBM Cloud Orchestrator ensures that the users in one project cannot access another project. User A can deploy virtual machines that connect only to Network A1 or Network A2. User B can deploy virtual machines that connect only to Network B1 or Network B2.

You can repeat the steps in this procedure to create additional projects and networks.
Neutron quota configuration:

You must update the Neutron quota configuration on each network node to accommodate the intended usage pattern. The default values might not be sufficient for clouds with more than 50 virtual machines.

On the Neutron node, edit the quota section of /etc/neutron/neutron.conf.

[QUOTAS]
# resource name(s) that are supported in quota features
quota_items = network,subnet,port

# default number of resource allowed per tenant, minus for unlimited
default_quota = -1

# number of networks allowed per tenant, and minus means unlimited
quota_network = 10

# number of subnets allowed per tenant, and minus means unlimited
quota_subnet = 10

# number of ports allowed per tenant, and minus means unlimited
quota_port = 50

# number of security groups allowed per tenant, and minus means unlimited
quota_security_group = 10

# number of security group rules allowed per tenant, and minus means unlimited
quota_security_group_rule = 100

# default driver to use for quota checks
quota_driver = neutron.db.quota_db.DbQuotaDriver

Initializing the IaaS toolkit database

Before using the IaaS toolkit, you must initialize the database.

To initialize the IaaS toolkit database, see the IaaS configuration instructions in Installing and configuring.

Performing advanced configuration tasks

After successful installation of IBM Cloud Orchestrator, you can perform advanced configuration tasks that help you better manage the product.

Configuring OpenStack to support resource overcommitment

IBM Cloud Orchestrator supports memory and CPU overcommitment for VMware and KVM regions.

About this task

In the /etc/nova/nova.conf file on the region server, use the cpu_allocation_ratio configuration option to specify the virtual CPU to physical CPU allocation ratio, and use the ram_allocation_ratio configuration option to specify the virtual RAM to physical RAM allocation ratio.

After you change the /etc/nova/nova.conf file, you must restart the nova services.
Deploying multiple Cinder nodes as Storage Nodes

You can deploy multiple Cinder nodes as Storage Nodes by customizing the Deployment Topology Templates.

**Procedure**

1. Setup a Cinder volume service on the existing nodes, take the KVM Compute Node for example:
   
   Modify the chef-runlist by adding two roles `role[os-block-storage-scheduler], role[os-block-storage-volume]`, like in the following sample, then follow the Cloud-Deployer documentation to update templates and jobs. In this example, all the KVM Compute Nodes in the cloud deployment have a Cinder volume service enabled:

   ```json
   "kvm_compute": {
     "Type": "IBM::SCO::Node",
     "Properties": {
       "Address": "COMPUTE_ADDR",
       "User": "root",
       "Password": "passw0rd",
       "KeyFile": "/home/heat/.ssh/heat_key"
     },
     "Metadata": {
       "chef-runlist": "role[kvm-compute],role[os-block-storage-scheduler],role[os-block-storage-volume]",
       "order": 3,
       "multiple": true
     }
   }
   ```

2. Setup a Cinder volume service on the new nodes:

   Follow the Cloud-Deployer documentation to define new resource name in template files, modify `chef-runlist` of the resource name by adding `role[os-block-storage-scheduler], role[os-block-storage-volume]`. After the nodes are deployed with the newly-defined resource name, the Cinder volume service is enabled.

**Advanced configuration on VMware region**

You can perform an advanced configuration of a VMware region.

**Enabling SRDS and Resource Pool**

If you want to adopt the following advanced features on a VMware region, you must update the `nova.conf` on the VMware Region Server and restart the `openstack-nova-compute` service.

**Storage cluster (SDRS):**

To support VMware Storage Clusters, edit the following settings are under the `[vmware]` section in `nova.conf`:

- `datastore_cluster_name`: is the Name of a VMware Datastore Cluster (StoragePod name). The default value is `None`.
- `use_sdrs`: specifies whether or not a driver must attempt to call DRS when cloning a virtual machine template. The default value is `False`.

**Note:** This feature is only supported when you boot a virtual machine from template.

You can use the following `extra_specs` of flavor to override the above configurations when you boot new virtual machines. To set or unset the flavor `extra_specs`, use `nova flavor-key` command.

```
vmware:datastore_cluster_name
```

Set this to override the datastore cluster name in `nova.conf`.  

---

Chapter 2. Installing 117
vmware:use_sdrs
Set this to override the use_sdrs in nova.conf.

Resource Pool:
To support the VMware resource pool, you must add the following
property under the [vmware] section in nova.conf:

resource_pool=<cluster_name>:<resource_pool_name>

where <cluster_name> is the name of the cluster. It is optional. If you
specify a cluster name and a resource pool name, the resource pool under
the cluster is the target to deploy the virtual machine. To select the
resource pool for deployment if there are multiple resource pools within a
cluster, there is a separate aggregated host defined for each resource pool.
This property is useful in scenarios in which you want to target a specific
datastore for deployment.

Connecting to multiple clusters
The installation only supports connecting one cluster. When you set up a VMware
region, if you want to connect to multiple clusters, you can do so manually. Each
cluster ends up in a new availability zone in the same region.

About this task
Because one openstack-nova-compute service can only connect to one cluster in
OpenStack Icehouse, you must create a new openstack-nova-compute service to
connect your new cluster. The new cluster results in a new host aggregate in a new
availability zone. In this procedure, nova-new-cluster is the name of the new
availability zone and new-aggregate-host is the name of the new aggregate host.
When you configure the multiple nova compute services to connect to multiple
vCenters or clusters, you also must configure the VMware discovery to connect to
these vCenters or clusters, for the details refer to "Configuring vmware-discovery
for multiple vCenters or cluster/resource pool" on page 126.

Procedure
1. Create the aggregate host and associate it with the new availability_zone with
   the command:
   nova aggregate-create new-aggregate-host nova-new-cluster
   nova aggregate-add-host new-aggregate-host nova-new-cluster
2. Create a copy of /etc/nova/nova.conf and name it /etc/nova/nova-new-
   SERVICE.conf.
3. Modify the /etc/nova/nova-new-SERVICE.conf to set:
   default_availability_zone = nova-new-cluster
   default_schedule_zone = nova-new-cluster
   host = nova-new-cluster
   host_ip = <your vCenter IP or host name>
   cluster_name = <the name of the additional cluster in your vCenter>
   resource_pool = <cluster_name>:<resource_pool_name-within-the-cluster>

   Note: Since we only support to connect to one resource pool in one nova
   compute service, if you want to connect to resource pool here, you also need
   specify the resource_pool under the cluster_name. For the resource pool
   configuration detail, see "Enabling SRDS and Resource Pool" on page 117.

   Note: If you run this new nova compute service on another separate machine,
   you must not change the default value of default_availability_zone,
   default_schedule_zone, and host.
4. Create a copy of /etc/init.d/openstack-nova-compute and name it
   /etc/init.d/openstack-nova-compute-new-SERVICE. As an example, ensure the
   openstack-nova-compute-new-SERVICE has ownership of root:root (chown

5. Modify the /etc/init.d/openstack-nova-compute-new-SERVICE to set:

   ```
   suffix=compute-new-SERVICE
   prog=openstack-nova-$suffix
   exec=/usr/bin/nova-compute
   config=/etc/nova/nova-new-SERVICE.conf
   pidfile=/var/run/nova/nova-$suffix.pid
   logfile=/var/log/nova/$suffix.log
   -e /etc/sysconfig/$prog ] & & /etc/sysconfig/$prog
   lockfile=/var/lock/subsys/$prog
   ```

6. Run the following commands to start services:
   chkconfig openstack-nova-compute-new-SERVICE on
   /etc/init.d/openstack-nova-compute-new-SERVICE start

7. Verify if the new compute service is up and running. Run the nova
   availability-zone-list command to check if the new availability zone which
   is named nova-new-cluster is shown.

   Verify if the new compute service is up and running. Run the command to
   check if the new availability zone which named <your-availability-zone-
   name> is shown:

   ```
   nova availability-zone-list
   ```

   ```
   +---------------------------------------------+-----------------------------------------------------------+
   | Name | Status |
   +---------------------------------------------+-----------------------------------------------------------+
   | internal | available | # In case you didn't change the name of the cluster
   | - <your-local-host-name> | | |
   |   - nova-conductor | enabled :-) 2014-08-07T05:15:44.766879 | |
   |   - nova-vmware | enabled :-) 2014-08-07T05:15:51.017709 | |
   |   - nova-consoleauth | enabled :-) 2014-08-07T05:15:49.413705 | |
   |   - nova-cert | enabled :-) 2014-08-07T05:15:47.481551 | |
   |   - nova-scheduler | enabled :-) 2014-08-07T05:15:47.36521 | |
   | nova | available | |
   | - <your-local-host-name> | | |
   |   - nova-compute | enabled :-) 2014-08-07T05:15:43.274219 | |
   | <your-availability-zone-name> | available | |
   | - <your-host-name> | | |
   |   - nova-compute | enabled :-) 2014-08-07T05:15:44.309888 | |
   +-----------------------------------------------+-----------------------------------------------------------+
   ```

   **Attention:** You can add more `cluster_name` properties in nova.conf instead of
   creating the new host aggregate and the new availability zone, but this results
   in having two or more clusters in the same availability zone which is not a
   good practice from a VM placement point of view.

**Connecting to multiple vCenters**

The installation only supports connecting to one vCenter. When you have set up a
VMware region, if you want to connect to multiple vCenters, you can perform the
following steps. Each cluster ends up in a new availability zone in the same region.

**About this task**

When you configure the multiple nova compute services to connect to multiple
vCenters or clusters, you also must configure the VMware discovery to connect to
these vCenters or clusters, for the details refer to "Configuring vmware-discovery
for multiple vCenters or cluster/resource pool" on page 126.
Procedure

1. For nova-compute:

   Since one openstack-nova-compute service can only connect to one vCenter in OpenStack Icehouse, we must create a new openstack-nova-compute service to connect your new vCenter.

   a. Create aggregate:

      nova aggregate-create <your-aggregate-name> <your-availability-zone-name>

   b. Create a copy of /etc/nova/nova.conf and name it /etc/nova/nova-new-SERVICE.conf as an example, ensure the nova-new-SERVICE.conf has ownership of 'nova:nova' (chown nova:nova /etc/nova/nova-new-SERVICE.conf).

   c. Modify the /etc/nova/nova-new-SERVICE.conf to set:

      [DEFAULT]
      default_availability_zone = <your-availability-zone-name>
      # your-availability-zone-name should be different from the first vCenter configuration
      default_schedule_zone = <your-availability-zone-name>
      host = <your-host-name>
      # <your-host-name could not be host name of VMware region server to
      # avoid conflict with first VCenter configuration

      [vmware]
      host_ip = <your new vCenter IP or hostname>
      cluster_name = <the cluster name in your vCenter>
      # To use more than one cluster, simply add multiple cluster_name
      # lines in nova.conf with the appropriate cluster name.
      resource_pool = <cluster_name>:<resource_pool_name-within-the-cluster>

   Note: Since we only support to connect to one resource pool in one nova compute service, if you want to connect to resource pool here, you also need specify the resource_pool under the cluster_name. For the resource pool configuration detail, see "Enabling SRDS and Resource Pool" on page 117.

   Note: If you run this new nova compute service on another separate machine, you do not need to change the default value of default_availability_zone, default_schedule_zone, host.

   d. Create a copy of /etc/init.d/openstack-nova-compute and name it /etc/init.d/openstack-nova-compute-new-SERVICE as an example ensure the openstack-nova-compute-new-SERVICE has ownership of 'root:root' (chown nova:nova /etc/init.d/openstack-nova-compute-new-SERVICE).

   e. Modify the /etc/init.d/openstack-nova-compute-new-vCenter to set:

      config="/etc/nova/nova-new-SERVICE.conf"
      pidfile="/var/run/nova/nova-$suffix-new-SERVICE.pid"
      logfile="/var/log/nova/nova-$suffix-new-SERVICE.log"
      prog=openstack-nova-compute-new-SERVICE
      lockfile=/var/lock/subsys/$prog

   f. Run the following commands to start services:

      chkconfig openstack-nova-compute-new-SERVICE on

      /etc/init.d/openstack-nova-compute-new-SERVICE start

   Associate the aggregate with the host and availability_zone you set in the step c with the following command:

      nova aggregate-add-host <your-aggregate-name> <your-host-name>
g. Verify if the new computer service is up and running. Run the following command to check if the new availability zone which named <your-availability-zone-name> is shown:
nova availability-zone-list

| Name | Status |
| +---------------------------------------------+-----------------------------------------------------------|
| internal | available |
| - <your-local-host-name> | enabled :-) 2014-08-07T05:44.766879 |
| - nova-conductor | enabled :-) 2014-08-07T05:51.017709 |
| - nova-vmware | enabled :-) 2014-08-07T05:49.413705 |
| - nova-consoleauth | enabled :-) 2014-08-07T05:47.481551 |
| - nova-cert | enabled :-) 2014-08-07T05:47.736521 |
| - nova-scheduler | enabled :-) 2014-08-07T05:44.309888 |
| nova | available |
| - <your-local-host-name> | enabled :-) 2014-08-07T05:43.274219 |
| - nova-compute | available |
| <your-availability-zone-name> | enabled :-) 2014-08-07T05:47.36521 |
| - <your-host-name> | enabled :-) 2014-08-07T05:44.309888 |
| - nova-compute | enabled :-) 2014-08-07T05:44.309888 |

h. Run the following command to check if the new cluster is shown in hypervisor list:
nova hypervisor-list

Check log file /var/log/nova/compute-new-SERVICE to see if there are any errors.

```
nova aggregate-create <your-aggregate-name> <your-availability-zone-name>
```

2. For nova-network:
If you are using nova-network instead of neutron, you must follow the following procedure to create a nova network service for each new nova compute service:


b. Modify the /etc/init.d/openstack-nova-network-new-SERVICE to set:
```
config="/etc/nova/nova-new-SERVICE.conf"
pidfile="/var/run/nova/nova-$suffix-new-SERVICE.pid"
logfile="/var/log/nova/$suffix-new-SERVICE.log"
prog=openstack-nova-network-new-SERVICE
lockfile=/var/lock/subsys/$prog
```

Note: Ensure that you are using the same nova configuration file with the pair nova compute service

c. Run the following commands to start the services:
```
chkconfig openstack-nova-network-new-SERVICE on
/etc/init.d/openstack-nova-network-new-SERVICE start
```

d. Verify if the new network service is up and running. Run the following command to check if the new network service under the internal availability zone is shown:
nova availability-zone-list

| Name | Status |
| +---------------------------------------------+-----------------------------------------------------------|
| internal | available |
| - <your-local-host-name> | enabled :-) 2014-08-07T05:44.766879 |
| - nova-conductor | enabled :-) 2014-08-07T05:51.017709 |
| - nova-vmware | enabled :-) 2014-08-07T05:49.413705 |
| - nova-consoleauth | enabled :-) 2014-08-07T05:47.481551 |
| - nova-cert | enabled :-) 2014-08-07T05:47.736521 |
| - nova-scheduler | enabled :-) 2014-08-07T05:44.309888 |
| nova | available |
| - <your-local-host-name> | enabled :-) 2014-08-07T05:43.274219 |
| - nova-compute | available |
| <your-availability-zone-name> | enabled :-) 2014-08-07T05:47.36521 |
| - <your-host-name> | enabled :-) 2014-08-07T05:44.309888 |
| - nova-compute | enabled :-) 2014-08-07T05:44.309888 |
Run the command to check if the new cluster is shown in hypervisor list.

3. For cinder volume:

Multiple vCenters support for cinder-volume. Since one openstack-cinder-volume service can only connect to one vCenter in OpenStack Icehouse, we must create a new openstack-cinder-volume service to connect your new vCenter.

a. Create a copy of `/etc/cinder/cinder.conf` and name it `/etc/cinder/cinder-new-vCenter.conf` as an example.

b. Modify the `/etc/cinder/cinder-new-vCenter.conf` to set:

```bash
classic_application = cinder-volume
storage_availability_zone = <your-availability-zone-name>
# your-availability-zone-name should be different from the first vCenter configuration
default_availability_zone = <your-availability-zone-name>
vmware_volume_folder = <your-volume-folder>
# your-volume-folder should be different from the first vCenter configuration,
# the default configuration for first is cinder-volumes
host = <your-host-name>
# your-host-name could not be host name of VMware region server to avoid
# conflict with first vCenter configuration
vmware_host_ip = <your new vCenter IP>
vmware_host_username = <your username to new vCenter>
vmware_host_password = <your password to new vCenter>
```

Note: If you run this new cinder volume service on another separate machine, you do not need to change the default value of `storage_availability_zone`, `default_schedule_zone`, `host`.

c. Create a copy of `/etc/init.d/openstack-cinder-volume` and name it `/etc/init.d/openstack-cinder-volume-new-vCenter` as an example.

d. Modify the `/etc/init.d/openstack-cinder-volume-new-vCenter` to set:

```bash
config="/etc/cinder/cinder-new-vCenter.conf"
pidfile="/var/run/cinder/cinder-$suffix-new-vCenter.pid"
logfile="/var/log/cinder/cinder-$suffix-new-vCenter.log"
prog=openstack-cinder-volume-new-vCenter
lockfile=/var/lock/subsys/$prog
```

e. Run the following commands to start the services:

```bash
chkconfig openstack-cinder-volume-new-vCenter on
/etc/init.d/openstack-cinder-volume-new-vCenter start
```

f. Verify if the new cinder service is up and running. Run the following command to check if the new availability zone which named 'cinder-new-vCenter is shown:
cinder availability-zone-list

Run the following command to check if the new cinder-volume service is shown in the service list:

cinder service-list

Check log file /var/log/cinder/volume-new-vCenter.log to see if there are any errors.

Multiple vCenters support for neutron:
Neutron has no particular configuration for multiple vCenters. For information about creating networks for neutron, see “Adding Neutron networks” on page 110. Pay attention that a port is created manually after neutron net-create command.

Work with the new vCenter:
a. Create a volume with the specified availability zone:
cinder create --display-name <My volume> <size> --availability-zone <My AZ>
b. Deploy the instance with the specified availability zone:
nova boot <My_instance> --image <My_image> --flavor <My_flavor> --availability-zone <My_AZ> --nic_net

Port group validation when using nova network
You can validate a port group when using a nova network.

Add attribute, ignore_vswitch_validation, under the [vmware] section in nova.conf. The default is False. If True, then do not check if the port group and the vlan interface are associated on the same virtual switch.

Add attribute, ignore_portgroup_create, under the [vmware] section in nova.conf. The default is False. If True, then do not create the port group when using nova network.

Enable the datastore random selection when booting a virtual machine
You can enable the datastore random selection when booting a virtual machine.

Add attribute, random_datastore, under the [vmware] section in nova.conf. The default is False. If True, then it randomly choses a datastore from the available list.

Configuring vmware-discovery
The vmware-discovery process discovers existing virtual machines, templates, and port groups in VMware.

Before you begin
1. The vmware-discovery configuration file is /etc/vmware-discovery.conf.
2. The vmware-discovery service is nova-discovery.

About this task
The vmware-discovery code is installed in a VMware region in IBM Cloud Orchestrator. To configure vmware-discovery, complete the following procedure.

Restriction: IBM Cloud Orchestrator V2.4 does not support the creation of a volume based on a discovered image in a VMware Region.
**Procedure**

1. Log on to the VMware Region Server as a user with root or sudo access.
2. Edit the `/etc/vmware-discovery.conf` configuration file.

The following table describes the parameters in the `/etc/vmware-discovery.conf` file:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>allow_instance_deletion</code></td>
<td>The default value is false. If set to True, the vmware-discovery process can delete the instances from OpenStack Nova that are deleted from the vCenter directly.</td>
</tr>
<tr>
<td><code>allow_template_deletion</code></td>
<td>The default value is false. If set to True, the vmware-discovery process can delete the templates from OpenStack Glance that are deleted from the vCenter directly.</td>
</tr>
<tr>
<td><code>auth_url</code></td>
<td>The Keystone public endpoint. You can find this value in the <code>/root/openrc</code> file or the <code>/root/keystonerc</code> file.</td>
</tr>
<tr>
<td><code>keystone_version</code></td>
<td>This value depends on the value of the <code>auth_url</code> parameter. If the <code>auth_url</code> value ends with <code>/v2.0</code>, <code>keystone_version</code> is <code>v2.0</code>. Otherwise, <code>keystone_version</code> is <code>v3</code>.</td>
</tr>
<tr>
<td><code>admin_tenant_name</code></td>
<td>The administration project. The default value is <code>admin</code>.</td>
</tr>
<tr>
<td><code>admin_user</code></td>
<td>The Cloud Administrator user. The default value is <code>admin</code>.</td>
</tr>
<tr>
<td><code>admin_password</code></td>
<td>The password of the specified Cloud Administrator user.</td>
</tr>
<tr>
<td><code>discovery_driver</code></td>
<td>The full class name for the driver for the VMware Discovery Service (string value). The value is <code>vmware.nova.driver.virt.vmware.driver.VMwareVCSynchingDriver</code>.</td>
</tr>
<tr>
<td><code>discovery_manager</code></td>
<td>The discovery driver. The value is <code>vmware.nova.driver.compute.manager.VMwareDiscoveryManager</code>.</td>
</tr>
<tr>
<td><code>staging_project_name</code></td>
<td>The project to which all discovered instances belong.</td>
</tr>
<tr>
<td><code>staging_user</code></td>
<td>The user to whom all discovered instances belong.</td>
</tr>
<tr>
<td><code>image_periodic_sync_interval_in_seconds</code></td>
<td>The image periodic sync interval specified in seconds. This interval is the time from the end of one successful image periodic sync operation to the start of the next. The default value is 300.</td>
</tr>
<tr>
<td><code>instance_prefix</code></td>
<td>The prefix for all discovered instances.</td>
</tr>
<tr>
<td><code>instance_sync_interval</code></td>
<td>The instance periodic sync interval specified in seconds. The default value is 20. The minimum value is 10.</td>
</tr>
<tr>
<td><code>image_limit</code></td>
<td>The maximum number of images to return. The default value is 500. If your OpenStack has more than 500 images, this limit should be increased to include all images.</td>
</tr>
<tr>
<td><code>image_sync_retry_interval_time_in_seconds</code></td>
<td>The time in seconds between image sync retry attempts if an error was encountered during an image sync operation. The default value is 60.</td>
</tr>
<tr>
<td><code>longrun_loop_interval</code></td>
<td>The minimum delay interval and initial delay in seconds for long run tasks. The default interval value is 7, and the default delay value is 10.</td>
</tr>
<tr>
<td><code>longrun_initial_delay</code></td>
<td>The prefix for all created flavors for the discovered instances.</td>
</tr>
<tr>
<td><code>full_instance_sync_frequency</code></td>
<td>The number of instance sync intervals between full instance syncs. At most intervals, only the instances that are known to be out of sync are synchronized. However, after the number of intervals specified by this parameter, all instances are synchronized. The default value is 30.</td>
</tr>
</tbody>
</table>
Table 11. Parameters in the /etc/vmware-discovery.conf configuration file (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>vm_ignore_list</td>
<td>The virtual machines that should be ignored by the discovery process.</td>
</tr>
<tr>
<td>vmware_default_image_name</td>
<td>The default image for the discovered instances.</td>
</tr>
<tr>
<td>physical_network_mappings</td>
<td>The virtual switch in your vCenter.</td>
</tr>
<tr>
<td>port_group_filter_list</td>
<td>The port group name that you want to use. If left blank, the process discovers all port groups.</td>
</tr>
<tr>
<td>portgroup_sync_interval</td>
<td>The port group periodic sync interval specified in seconds. The default value is 300. The minimum value is the value of the instance_sync_interval parameter.</td>
</tr>
<tr>
<td>target_region</td>
<td>The name of the region where the discovered resources are registered. The default value is the name of the current region.</td>
</tr>
<tr>
<td>template_sync_interval</td>
<td>The template periodic sync interval specified in seconds. The default value is 300. The minimum value is the value of the instance_sync_interval parameter.</td>
</tr>
</tbody>
</table>

**Important:** Do not change the default values of the other parameters in the /etc/vmware-discovery.conf configuration file.

3. Start the vmware-discovery service:
   
   ```
   service nova-discovery start
   ```

   **Note:**
   - This service applies to two configuration files by default: /etc/nova/nova.conf and /etc/vmware-discovery.conf. The vCenter information is in the /etc/nova/nova.conf file. You can assign a different configuration file by specifying the --config-file your_configuration_file parameter. You can add this parameter in the /etc/init.d/nova-discovery service script, as described in “Configuring vmware-discovery for multiple vCenters or cluster/resource pool” on page 126.
   - If you want to stop the vmware-discovery service, run the following command:
     ```
     service nova-discovery stop
     ```

4. Log for vmware-discovery:
   
   You can find the discovery log in the /var/log/nova/discovery.log file.

**Results**

The vmware-discovery process is configured and started after installation in the VMware region. The default value for the staging project and the staging user is admin. You can now manage the discovered resources as described in “Managing resources” on page 306.
Configuring `vmware-discovery` for multiple vCenters or cluster/resource pool

It is possible to connect more than one vCenter or cluster in one region.

About this task

Refer to "Connecting to multiple vCenters" on page 119.

Restriction: For a VMware region that connects to multiple vCenters, if images with the same name exist on different vCenters, only one of these images can be discovered in the VMware region. Use command `nova image-show <image-id>` to find the image location on each vCenter.

Restriction: IBM Cloud Orchestrator 2.4 supports one VMware region connect to multiple vCenter, and all of vCenter must locate in same network with VMware neutron node. IBM Cloud Orchestrator 2.4 supports one KVM region connect to multiple computer nodes, and all of the computer nodes must locate in same network with the KVM neutron node.

Procedure

1. Create the configuration files for a certain vCenter or cluster:
   a. Configuration file for the `openstack-nova-compute`:
      By default, the VMmware discovery will load the VMware related information from `/etc/nova/nova.conf`. As you already connected to the other vCenter or cluster/resource pool, there should be a configuration file for this vCenter VMware information which is applied by the related `openstack-nova-compute` service. It is usually under `/etc/nova/`. Let's take it as `/etc/nova/nova-2.conf` for example, ensure the `vmware-discovery-2.conf` has ownership of 'nova:nova' (chown nova:nova /etc/nova/vmware-discovery-2.conf).
   
   b. Configuration file for the `vmware-discovery`:
      Customer could copy the `/etc/vmware-discovery.conf` and then modify it as the discovery configuration file for this new vCenter. Here takes it as `/etc/vmware-discovery-2.conf` for example:
      `cp /etc/vmware-discovery.conf /etc/vmware-discovery-2.conf`

2. Create the service file for a certain vCenter or cluster:
      `cp /etc/init.d/nova-discovery /etc/init.d/nova-discovery-2`

   b. Modify the new service file.
      1) `/etc/init.d/nova-discovery-2`
      
      to:

      ```
      suffix=discovery-new-SERVICE
      prog=nova-$suffix
      pidfile="/var/run/nova/nova-$suffix.pid"
      logfile="/var/log/nova/$suffix.log"
      [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog
      lockfile="/var/lock/subsys/$prog"
      
      Apply the new configuration files:
      daemon --user nova --pidfile $pidfile "$exec --logfile $logfile >&/dev/null & echo \$! > $pidfile"
      ```

IBM Cloud Orchestrator 2.4: User's Guide
to;

d daemon --user nova --pidfile $pidfile "exec --config-file /etc/vmware-discovery-2.conf --config-file /etc/nova/nova-2.conf \ --logfile $logfile &>/dev/null & echo $! > $pidfile"

3. Start the new service:

/etc/init.d/nova-discovery-2 start
chkconfig nova-discovery-2 on

Configuring OpenStack to support thin provisioning
You can configure OpenStack to support thin provisioning.

About this task
Thin provisioning:
Thin provisioning is inherited from the template. If the template was created with thin provisioned disk then instances deployed out of that will be thin provisioned

Linked clones:
By default IBM Cloud Orchestrator uses linked clones. To change this behavior you must modify on the region server /etc/nova/nova.conf setting use_linked_clone to false. Then, restart nova services.

Removing a KVM compute node
To remove a KVM compute node from IBM Cloud Orchestrator, perform the following steps.

Procedure
1. List the services on all the compute nodes:

On the Region Server, run: ~/openrc && nova service-list

<table>
<thead>
<tr>
<th>Binary</th>
<th>Host</th>
<th>Zone</th>
<th>Status</th>
<th>State</th>
<th>Updated_at</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova-cells</td>
<td>rs-135-3</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-10-09T06:47:41:699503</td>
</tr>
<tr>
<td>nova-cert</td>
<td>rs-135-3</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-10-09T06:47:48:464156</td>
</tr>
<tr>
<td>nova-compute</td>
<td>rs-5-blade-6</td>
<td>nova</td>
<td>enabled</td>
<td>up</td>
<td>2013-10-09T06:47:42:402192</td>
</tr>
<tr>
<td>nova-conductor</td>
<td>rs-135-3</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-10-09T06:47:49:713611</td>
</tr>
<tr>
<td>nova-network</td>
<td>rs-5-blade-6</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-10-09T02:47:32:317892</td>
</tr>
<tr>
<td>nova-consoleauth</td>
<td>rs-135-3</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-10-09T06:47:49:928091</td>
</tr>
<tr>
<td>nova-scheduler</td>
<td>rs-135-3</td>
<td>internal</td>
<td>enabled</td>
<td>up</td>
<td>2013-10-09T06:47:49:927776</td>
</tr>
</tbody>
</table>

2. Disable the nova compute and nova network service on the compute node you want to remove. On the Region Server, run: nova-manage service disable --host=<host> --service=<services>

For example, if you want to remove node rs-5-blade-6, run the following commands:

nova-manage service disable --host=rs-5-blade-6 --service=nova-network
nova-manage service disable --host=rs-5-blade-6 --service=nova-compute

3. If you want to remove the compute node completely, you must manually remove it from the database, create a file drop_node.sql with following contents:

connect to openstack user <dbuser> using <dbpassword>
delete from compute_node_stats where compute_node_id in
(select id from compute_nodes
  where hypervisor_hostname='rs-5-blade-6')
delete from compute_nodes where hypervisor_hostname='rs-5-blade-6'
delete from services where host='rs-5-blade-6'

Log on to central server(share db) / region server(non-shared db) with user db2inst1 and run the sql script:
su - db2inst1
db2 -tf drop_node.sql

4. (Optional, this step is required before re-adding a removed compute back to the region server). On the compute node, clean the OpenStack packages. Find the packages needed to remove: rpm -qa | grep "*openstack*". Remove the packages using the command: rpm -e <package name>

5. (Optional, this step is required before re-adding a removed compute back to the region server). Restore the network configuration of the compute node. Remove the bridge which created by the IBM Cloud Orchestrator installer and reassign the IP back to ethX interface.

6. Delete the node from the Deployment Server by using the ds node-delete command.

What to do next

The availability zone will be removed from the OpenStack region after all the compute nodes that used this availability zone are removed. You can run the command nova availability-zone-list to find the compute nodes that use the availability zone.

Removing a region

To remove a region from the IBM Cloud Orchestrator environment, perform the following steps.

Procedure

1. The IBM Cloud Orchestrator administrator must delete all the instances on the region:
   - Log on to the Self-service user interface and click ASSIGNED RESOURCES.
   - Review each of the instances and delete the instances that belong to the region that you want to remove.

2. On Central Server 2, update the Keystone entries that relate to the region that you want to remove, for example, RegionOne:
   source /root/keystonerc
   keystone endpoint-list | grep RegionOne | cut -d '|' -f2 | while read endpoint;
   do keystone endpoint-delete $endpoint;
   done

3. Update the IBM Cloud Orchestrator environment as follows:
   - Delete the region cloud groups:
     a. Click PATTERNS > Deployer Configuration > Cloud Groups.
     b. Select the cloud group for the region you are deleting and click Discover.
     c. Select each cloud group related to the region and click Delete.
   - Delete the region IP group:
     a. Click PATTERNS > Deployer Configuration > IP Groups.
     b. Select the required IP group and click Delete.

4. To check if the region has been successfully removed, perform the following steps:
   a. Be sure that no endpoints are found in Keystone. Run the following commands on Central Server 2:
      source /root/keystonerc
      keystone endpoint-list | grep RegionOne

      where RegionOne is the region you removed. No output should be returned.
b. Be sure there are no cloud groups related to this region in IBM Cloud Orchestrator. Log in to the Self-service user interface and click PATTERNS > Deployer Configuration > Cloud Groups. There should be no cloud groups related to the region.

5. Remove the region environment from IBM Cloud Orchestrator tool:
   a. Go to the /opt/ibm/orchestrator/scoorchestrator directory.
   b. Remove the SCOEnvironment_<removed_region>.xml file.
   c. Run the SCOEnvironment_update.py script.
   d. Check that all the node information of the removed region is not present anymore in the SCOEnvironment_fulltopo.xml file.
   e. Run the SCOrchestrator.py --status command to check if the status of the removed region is gone.

### Configuring LDAP authentication

Configure IBM Cloud Orchestrator to use an LDAP server for user authentication.

By default, IBM Cloud Orchestrator authenticates users against the OpenStack Keystone database that is part of the product. You can configure IBM Cloud Orchestrator to authenticate against an external corporate directory (such as LDAP or Active Directory) by enabling IBM Cloud Orchestrator to pass an authentication request to the corporate directory before passing the request to Keystone.

The LDAP authentication support provided by IBM Cloud Orchestrator enables users defined in LDAP to logon to IBM Cloud Orchestrator by specifying the LDAP credentials (user name and password) if configured correctly. However, users defined locally in OpenStack Keystone are not populated into LDAP.

If you have a single corporate directory server which contains all the users irrespective of which domain they are in, then follow the procedure described in "Configuring LDAP authentication independently of the domain."

In a multidomain configuration, IBM Cloud Orchestrator extends the pre-authentication capability to allow any corporate directory details to be specified on a domain-by-domain basis. In this mode, for any domain that requires separate corporate directory details, you can define a specific partial Keystone configuration file in addition to the main Keystone configuration file that is used by all the domains. To configure LDAP authentication for a specific domain, follow the procedure described in "Configuring LDAP authentication on a domain-by-domain basis" on page 133.

### Configuring LDAP authentication independently of the domain

You can configure LDAP authentication for IBM Cloud Orchestrator.

To configure LDAP authentication, you must already have an LDAP server set up. You must specify which LDAP attributes are used as part of the authentication process. Consider the following items:

- Which LDAP attribute is checked against the user name as part of authentication? The default is the common name (cn) attribute, but you can choose whatever suits your environment, for instance, email is a common alternative. Whatever you chose here, is what the user must type into the IBM Cloud Orchestrator login panel.
- Where are the user objects stored in the LDAP server? You must specify the root of the subtree that contains these user objects. They do not need to exists at the same level, but they do need to be all within one subtree.
• Does your LDAP server support anonymous query? If not, you need an LDAP user and password account that does have search capabilities, for the subtree that contains the user’s objects.

**Note:** The IBM Cloud Orchestrator administrator account admin already exists in the local OpenStack database. Make sure that there is not an LDAP account that also has the name admin, or this supersedes the local admin account.

After establishing these items, perform the following steps to configure LDAP authentication:

1. Make sure that the ldapauth.py file exists in your Keystone installation in the keystone/middleware directory.

2. The following properties can be defined in your Keystone configuration file to control LDAP authentication. Comment the existing [ldap] section before you add an [ldap_pre_auth] section. The keystone configuration file is, by default, to be found in /etc/keystone/keystone.conf on Central Server 2:

   ```
   [ldap_pre_auth]
   # The url of the corporate directory server
   url = ldap://localhost
   # The root of the tree that contains the user records
   user_tree_dn = cn=users,dc=example,dc=com
   # The property in the user record that will be checked against the username
   user_name_attribute = cn
   # In order to search for user records, we will try and use anonymous query.
   # If anonymous query is not available, then define the user and password
   # of an account that does have rights to do a search
   user = cn=admin,cn=users,dc=example,dc=com
   password = <admin_password>
   # Define this property if you want to customize the user id
   # which will be used if we automatically populate the user to keystone
   user_id_attribute = dn
   # By default if we fail to find a user in LDAP, we will then try and
   # find that user directly in keystone. If you don't want that to happen
   # then set pass_through to False
   pass_through = False
   ```

3. You can configure support for LDAP SSL or TLS. To use SSL, you must configure the underlying openldap client with the appropriate certificate details which are typically stored in /etc/openldap/ldap.conf, for example:

   ```
   TLS_CACERT /etc/openldap/certs/serverkey.cer
   TLS_REQCERT DEMAND
   ```

   With openldap configured, you can refer to your secure LDAP server by specifying 'url = ldaps://....' in keystone.conf.

   If you use TLS, then the certificate details are specified in the keystone.conf file itself:

   ```
   [ldap_pre_auth]
   tls_cacertfile = /path/to/certfile
   tls_cacertdir = /path/to/certdir
   tls_req_cert = demand
   use_tls = True
   ```

   `tls_cacertfile` and `tls_cacertdir` are not both required. If you use TLS, you must use the regular 'url = ldaps://....' connection specification (and not use ldaps).

4. The LDAP objects that are checked during authentication can be restricted using two variables in the configuration file. `user_objectclass` allows you to specify the class of objects to be checked and `user_filter` allows you to specify additional AND filters to be used in the objects being checked. For instance, the
following statement restricts authentication to objects of class person that have an attribute `memberOf` indicating membership to an existing OpenStack group in your corporate directory server:

```plaintext
[ldap_pre_auth]
user_objectclass = person
user_filter = (memberOf=CN=OpenStack,dc=ldap,dc=com)
```

5. In your Keystone paste configuration file, `keystone-paste.ini` by default, define a filter for this class:

```plaintext
[filter:ldapauth]
paste.filter_factory = keystone.middleware.ldapauth:LdapAuthAuthentication.factory
```

6. Comment the existing pipeline and add this line in the `keyston-paste.ini` file:

```plaintext
[pipeline:api_v3]
#pipeline = access_log sizelimit url_normalize token_auth admin_token_auth xml_body json_body simpletoken ec2_extension s3_extension service_v3
pipeline = access_log sizelimit stats_monitoring url_normalize token_auth admin_token_auth xml_body json_body simpletoken ldapauth autopop debug
```

This procedure enables authentication to occur against your corporate directory. However, a representation of an authenticated user must also be available in Keystone after authorization so that, for example, roles can be assigned to the user. While you can do this manually, or using an external directory synchronization tool, IBM Cloud Orchestrator provides you with an auto-population plug-in that can help you perform simple, automatic population of an LDAP user to Keystone when that user first authenticates successfully to LDAP. The plug-in only populates the minimal amount of information required into Keystone, that is, user name and user ID. Because the plug-in does not propagate the user password, these Keystone user entries do not allow the LDAP authentication step to be bypassed. They are only present to allow role assignment. To configure auto-population, you need to make the following choices:

- Which LDAP attribute is used to create a unique `user_id` within the OpenStack Keystone? This must be something that does not change over time, that is limited to 64 characters, and that does not contain blank space characters. The default is the distinguished name (DN) attribute, but you can choose whatever suits your environment, for example, the email attribute is a common alternative. It is acceptable, for example, to use email as both the username and the `user_id`.

- Which project must the user be given an initial role on as part of the auto-population? This project already needs to exist before the auto-population takes place.

**Note:** It is not recommended that you enable a situation where a user sometimes authenticates with LDAP and sometimes directly with Keystone, since unless these are carefully maintained, with matching `user_ids` and so forth, this can cause confusion about which is the master user record.

**Note:** The auto-population plug-in is not a replacement for a full directory synchronization capability. While the plug-in creates a user-record the first time a user authenticates, it does not, for example, delete that user record in keystone if the LDAP user record is subsequently deleted. Such a user would indeed no longer be able to log in to IBM Cloud Orchestrator, but the database cleanup is outside of the scope of the auto-population plug-in. If such a full capability is required, then the use of an external directory synchronization tool is recommended (for details refer to “Using external Directory Integration tools with LDAP authentication” on page 133).

1. Make sure the `autopop.py` file exists in your Keystone installation in the `keystone/middleware` directory.
2. The main job of the auto-population plug-in is to ensure that a user record is created in Keystone, so that subsequent role assignments can take place. However, optionally, the plug-in can grant an initial role to that user. This is achieved by specifying the project and role required in an [auto_population] section of the configuration file. First, you must specify the project, by providing either a project ID by defining default_project_id or a project name by defining default_project. If a project name is specified, then this is assumed to be in the same domain as the user who has just been authenticated, whereas a project ID is, by definition, unique across all domains. For example:

```
[auto_population]
default_project = test-project
```

instructs the auto population plug-in to assign the authenticated user the standard membership role in the project named test-project in the user’s domain. This project must already exist for this role assignment to take place. Additionally, you can ask for the plug-in to grant a further explicit role to the user on the project by specifying either a role name using a default_role or a role ID using default_role_id. Hence, the following gives an authenticated user both the membership role and the vm-manager role:

```
[auto_population]
default_project = test-project
default_role = vm-manager
```

**Note:** The roles assigned here only affect the assignments held within IBM Cloud Orchestrator and Keystone. They do not modify anything that is actually stored in LDAP. The plug-in does not attempt to propagate any roles stored explicitly within LDAP, although the use of an external directory synchronization tool can enable such a capability.

**Note:** default_project_id is the standard config file variable for setting the default project by ID.

3. In your Keystone paste configuration file, keystone-paste.ini by default, define a filter for this class:

```
[filter:autopop]
paste.filter_factory = keystone.middleware.autopop:AutoPopulation.factory
```

4. Add the autopop plug-in to the pipeline that you are using, by default this is the api_v3 pipeline. The plug-in autopop must come after the ldapauth filter:

```
[pipeline:api_v3]
pipeline = access_log sizelimit stats_monitoring url_normalize token_auth admin_token_auth xml_body json_body simpletoken ldapauth autopop debug stats_reporting ec2_extension s3_extension service_v3
```

5. In your Keystone policy configuration file, policy.json, modify the list_user_projects parameter to allow users to list the projects they own, like:

```
"identity:list_user_projects": "rule:admin_or_owner_or_domain"
```
Configuring LDAP authentication on a domain-by-domain basis

There are some additional steps to configure LDAP authentication in an IBM Cloud Orchestrator multidomain environment.

To specify corporate directory details on a domain-by-domain basis, perform the following steps in addition to the procedure described in "Configuring LDAP authentication independently of the domain" on page 129:

1. For each domain that needs its own corporate directory details, in the `<domain_config_files_dir>` directory, you must create a domain-specific configuration file with the following name:
   `keystone.<domain_name>.conf`

   Include only the `[ldap_pre_auth]` and `[auto_population]` sections in the domain-specific configuration file, within which you must define the specific configuration for the LDAP server for that domain. The options available are the same as those listed in topic "Configuring LDAP authentication independently of the domain" on page 129.

2. Add the following properties in your Keystone configuration file:
   ```
   [ldap_pre_auth]
   domain_specific_drivers_enabled = True
   domain_config_dir = <domain_config_files_dir>
   ```

   where `<domain_config_files_dir>` is the directory where the domain specific configuration files are stored.

Note: TLS certification details cannot currently be specified on a domain-by-domain basis.

Using external Directory Integration tools with LDAP authentication

An alternative approach to using the provided auto-populate function is to use a directory synchronization product, such as, for example, Tivoli Directory Integrator.

The configuration of this external capability depends on the specific product, however you must perform the following steps:

1. In the user table in the Keystone SQL database, insert a record for each valid LDAP or Active Directory that you want to use with the product. The structure of the user table is as follows:
   ```
   id character varying(64) NOT NULL
   name character varying(64) NOT NULL
   domain_id character varying(64) NOT NULL
   password character varying(128) NOT NULL
   enabled Boolean
   extra text
   ```

   where:
   - `id` Must be unique across all user entries in Keystone. For example, it might be the dn of the LDAP user.
   - `name` Is the user name that is used for authentication.
   - `domain_id` Must match the `domain_id` for the domain the user is in. If you have not created any domains, this is the default domain for which the `domain_id` is 'default'. If you are using multiple domains, then the `domain_id` must match the entry in the domains table.
password
This is not used for authentication, since that is done against the corporate directory, so this must be set to some randomly generated string to prevent the user from bypassing the corporate directory authentication.

enabled
Must be set to true.

extra Can be left blank.

2. If the directory synchronization tool is used to delete user entries from the Keystone database in response to a deletion in the corporate directory, then there are a number of tables that must be updated and the user entry described above must finally be deleted. Since a number of these tables use foreign keys to point back to the user table, it is important that you only delete the user entry itself once all the changes listed below have been made.

In the following tables you must delete any entry with a user_id that matches the ID from the user table:

Table: UserProjectGrant:
- user_id character varying(64) NOT NULL
- project_id character varying(64) NOT NULL
- data text

Table: UserDomainGrant:
- user_id character varying(64) NOT NULL
- domain_id character varying(64) NOT NULL
- data text

Table: UserGroupMembership:
- user_id character varying(64) NOT NULL
- group_id character varying(64) NOT NULL

If you are using an SQL server to store tokens, then to ensure that the user being deleted is denied access to the product as soon as possible, the TokenModel table must also be updated. If you are archiving expired tokens for auditing purposes, then set the valid field to False for any token record that matches the user_id. If you are not concerned about token archiving, you can simple delete any token records that match this user_id.

Table: TokenModel:
- id character varying(64) NOT NULL
- expires datetime
- extra text
- valid boolean
- user_id character varying(64) NOT NULL
- trust_id character varying(64) NOT NULL

If using memcache as a token store, to disable a deleted LDAP user as soon as possible, the memcache entries of tokens for that user must be deleted. The memcache token store has two types of entries, with the keys of:

'usertokens-<user_id>'
Contains a list of token IDs for the user with the ID specified by <user_id>

'tokens-<token_id>'
A specific token

To remove all tokens for a user, a memcache 'get' must be issued for the key of 'usertokens-<user_id>' with the user_id in question. The value for this entry
is a list of token IDs, and for each of these delete the corresponding memcache key of 'tokens-<token_id>'. Finally, delete the key for 'usertokens-<user_id>'.

**LDAP Keystone configuration file reference**

The connection details of your corporate directory are stored within the [ldap_pre_auth] and [auto-population] sections of either the keystone.conf file or a domain specific configuration file.

The supported options within each of the two section are as follows:

*Table 12. Configuration file options in ldap_pre_auth section*

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain_config_dir</td>
<td>The directory that contains any domain specific configuration files. This is only used if domain_specific_drivers_enabled is True.</td>
</tr>
<tr>
<td></td>
<td>Default /etc/keystone/domains</td>
</tr>
<tr>
<td></td>
<td>Example domain_config_dir = /etc/myconfigs</td>
</tr>
<tr>
<td>domain_specific_drivers_enabled</td>
<td>If True, then both the ldapauth and autopop plugins will look in the domain_config_dir for domain specific configuration files of the form keystone.&lt;domainname&gt;.conf, where &lt;domainname&gt;.conf is the name given to the domain when it was created via the IBM Cloud Orchestrator UI. Domain configuration files are only read when the keystone component of IBM Cloud Orchestrator is started (or restarted).</td>
</tr>
<tr>
<td></td>
<td>Default False</td>
</tr>
<tr>
<td>non_ldap_users</td>
<td>The local user list is not authenticated with LDAP. This option helps to improve the authentication performance for built-in users. The users defined in the list must already exist in local database, otherwise the authentication fails.</td>
</tr>
<tr>
<td></td>
<td>Default None</td>
</tr>
<tr>
<td></td>
<td>Example non_ldap_users = admin, demo, nova, neutron, cinder, glance, monitoring, domadmin, heat, test</td>
</tr>
<tr>
<td>pass_through</td>
<td>If True, then if a user record is not found in the corporate directory server that matches the specified user_name_attribute, then the authentication request will be retried against the internal keystone database directly.</td>
</tr>
<tr>
<td></td>
<td>Default True</td>
</tr>
<tr>
<td></td>
<td>Example pass_through = False</td>
</tr>
<tr>
<td><strong>password</strong></td>
<td>The password for the user account specified by <code>user</code> to enable searching within the corporate directory. This is unrelated to the actual user and the password that is being authenticated.</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>password = secret</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>tls_cacertdir</strong></th>
<th>The directory where TLS certificates are stored.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>tls_cacertdir = /etc/openldap/cacertdir</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>tls_cacertfile</strong></th>
<th>The TLS certification file itself.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>tls_cacertfile = /etc/openldap/cacertdir/server.cer</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>tls_req_cert</strong></th>
<th>The certificate requirement specification. This may be one of the following: 'never', 'demand' or 'allow'.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td><code>tls_req_cert = 'demand'</code></td>
</tr>
</tbody>
</table>

| **url**          | The URL of the corporate directory server. If using SSL, you must also configure the underlying openldap client with the certificate details. These are typically stored in `/etc/openldap/ldap.conf`, for example: |
|                 | `TLS_CACERT /etc/openldap/certs/serverkey.cer`                                                                                                                                                       |
|                 | `TLS_REQCERT ALLOW`                                                                                                                                                                                  |
|                 | With TLS, however, you configure these details in the keystore configuration file itself (see the options `tls_cacert` and `tls_req_cert` below), and use a regular `ldap://...` URL.                              |
|                 | **Default** None                                                                                                                                                                                     |
| **Example**      | `url = ldaps://www.ibm.com/myserver`                                                                                                                                                                |

If using SSL, then use `ldaps`, for example: `url = ldaps://www.ibm.com/mysecureserver`
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>The user account to be used for searching within the corporate directory. This is unrelated to the actual user and password that is being authenticated. If the corporate directory server supports anonymous query, then this must not be specified.</td>
<td>None</td>
<td>user = cn=root</td>
</tr>
<tr>
<td>user_attribute_name</td>
<td>This is now depreciated, but has the same meaning as user_name_attribute.</td>
<td>cn</td>
<td></td>
</tr>
<tr>
<td>user_filter</td>
<td>An additional filter (or filters) that will be ANDed into the search for the user_name_attribute. The filter must be in brackets, as in the example.</td>
<td>None</td>
<td>user_filter = (memberOf=cn=openstack,dc=root)</td>
</tr>
<tr>
<td>user_id_attribute</td>
<td>If auto-population is enabled, this is the attribute that will be used as the user_id when the keystone user record is created. A keystone user_id is limited to 64 characters and must not contain blank space characters and should be something that will not change for the lifetime of that user.</td>
<td>dn</td>
<td>user_id_attribute = mail</td>
</tr>
<tr>
<td>user_mail_attribute</td>
<td>If auto-population is enabled and it is required to propagate the email address into the Keystone database, then define the attribute within the corporate directory object to be used.</td>
<td>None</td>
<td>user_mail_attribute = mail</td>
</tr>
<tr>
<td>user_name_attribute</td>
<td>The attribute within each corporate directory object that will be compared against the user name in the authentication request.</td>
<td>cn</td>
<td>user_name_attribute = mail</td>
</tr>
</tbody>
</table>
### Table 12. Configuration file options in `ldap_pre_auth` section (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>user_objectclass</strong></td>
<td>The class of objects that will be searched to try and match the <code>user_name_attribute</code>.</td>
</tr>
<tr>
<td>Default</td>
<td>*</td>
</tr>
<tr>
<td>Example</td>
<td><code>user_objectclass = person</code></td>
</tr>
<tr>
<td><strong>user_tree_dn</strong></td>
<td>The root of the subtree in the corporate directory server that will be searched.</td>
</tr>
<tr>
<td>Default</td>
<td><code>cn=users,dc=example,dc=com</code></td>
</tr>
<tr>
<td>Example</td>
<td><code>user_tree_dn = cn=users,dc=root</code></td>
</tr>
<tr>
<td><strong>use_tls</strong></td>
<td>Indicates that the connection to the corporate directory should use TLS.</td>
</tr>
<tr>
<td>Default</td>
<td><code>False</code></td>
</tr>
<tr>
<td>Example</td>
<td><code>use_tls = True</code></td>
</tr>
</tbody>
</table>

### Table 13. Configuration file options in `auto-population` section

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>default_project</strong></td>
<td>The default project to be assigned to the new user being created, which means a member role will be given to this project for the user. The project must already exist. Note that this project must exist in the same domain as the user being created. If you want to assign the user a role in a project that is in a different domain, then use <code>default_project_id</code>.</td>
</tr>
<tr>
<td>Default</td>
<td><code>None</code></td>
</tr>
<tr>
<td>Example</td>
<td><code>default_project = project1</code></td>
</tr>
<tr>
<td><strong>default_project_id</strong></td>
<td>The default project ID to be assigned to the new user being created, which means a member role will be given to this project for the user. The project must already exist.</td>
</tr>
<tr>
<td>Default</td>
<td><code>None</code></td>
</tr>
<tr>
<td>Example</td>
<td><code>default_project_id = 87bb06e36a854c0b97c45b4e6dbf5e4</code></td>
</tr>
<tr>
<td><strong>default_role</strong></td>
<td>An additional role (specified by name) that will be granted to the newly created user on the default project (in addition to the member role).</td>
</tr>
<tr>
<td>Default</td>
<td><code>None</code></td>
</tr>
<tr>
<td>Example</td>
<td><code>default_role = vm-manager</code></td>
</tr>
</tbody>
</table>
Table 13. Configuration file options in auto-population section (continued)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default_role_id</td>
<td>An additional role (specified by ID) that will be granted to the newly created user on the default project (in addition to the member role).</td>
</tr>
<tr>
<td></td>
<td>Default: True</td>
</tr>
<tr>
<td></td>
<td>Example: default_role_id=34bc06e13a854c0b97c45b4e66bf5fe1</td>
</tr>
<tr>
<td>default_tenant_id</td>
<td>This is now depreciated, but has the same meaning as default_project_id.</td>
</tr>
<tr>
<td></td>
<td>Default: None</td>
</tr>
</tbody>
</table>

Changing the various passwords

You can change the password for IBM Cloud Orchestrator environment users.

The topic contains the following sections:
- "Changing the admin password"
- "Changing the bpm_admin and tw_admin passwords" on page 140
- "Changing the db2inst1 password" on page 140
- "Changing the bpmuser password" on page 142
- "Changing the vCenter password" on page 143
- "Changing the password for OpenStack service users" on page 143

Changing the admin password

The admin user is used across the following IBM Cloud Orchestrator components: IBM Cloud Orchestrator user interface, IaaS Gateway, and OpenStack.

1. Change the password in the IBM Cloud Orchestrator user interface by editing the admin user in the Users panel. Click Administration > Users. After you change the password, you cannot continue to work with the user interface until you complete step 4 of this procedure. Close your browser.

2. Log on to the Region Server and encrypt the new password using the /usr/bin/openstack-obfuscate <new password> command. Then log on to Central Server 2 and edit the /etc/iaasgateway/iaasgateway.conf file by changing the following line to the new password:
   
   password = <type the new admin encrypted password here>

3. Log on to Central Server 3 and restart the Workload Deployer service:
   - If you are not using the IBM Cloud Orchestrator high availability solution, run the following command:
     
     service iwd restart
   - If you are using the IBM Cloud Orchestrator high availability solution, log in to the System Automation Application Manager user interface and initiate a restart of the Workload Deployer resource.

4. Log on to Central Server 2 and edit the /root/keystonerc file to change the following line:
   
   export OS_PASSWORD=<old admin password>

   to:
   
   export OS_PASSWORD=<new admin password>
Changing the bpm_admin and tw_admin passwords

To change the bpm_admin password, follow these steps:

1. Change the password of bpm_admin in the WebSphere Application Server page:
   b. Select Users and Groups.
   c. Select bpm_admin.
   d. In the User Properties panel, set the password, confirm it, and click Apply.
   e. Change the following configuration files:
      1) On Central Server 4, create a backup copy of the following files:
         /opt/ibm/BPM/v8.5/profiles/DmgrProfile/properties/soap.client.props
         and:
         /opt/ibm/BPM/v8.5/profiles/Node1Profile/properties/soap.client.props
      2) Edit each of the soap.client.props files listed in step i to find the
         com.ibm.SOAP.loginUserid=bpm_admin entry, and update the associated
         com.ibm.SOAP.loginPassword entry to specify the new password as plain
         text:
         com.ibm.SOAP.loginUserid=bpm_admin
         com.ibm.SOAP.loginPassword=<type the new bpm_admin password here>
      3) Encrypt the password by running the following two commands:
         /opt/ibm/BPM/v8.5/bin/PropFilePasswordEncoder.sh
         /opt/ibm/BPM/v8.5/profiles/DmgrProfile/properties/soap.client.props
         com.ibm.SOAP.loginPassword
         /opt/ibm/BPM/v8.5/bin/PropFilePasswordEncoder.sh
         /opt/ibm/BPM/v8.5/profiles/Node1Profile/properties/soap.client.props
         com.ibm.SOAP.loginPassword
      4) Follow the additional configuration steps described in the Business
         Process Manager documentation.

2. To change the password of tw_admin, run the same procedure as described in
   Step 1.

Changing the db2inst1 password

The db2inst1 password must be changed in the operating system where the DB2
instance is installed and in the OpenStack configuration files.

Remember that db2inst1 is used for OpenStack and Workload Deployer database
connections.

1. To change the db2inst1 password in the operating system where DB2 is, log on
   the system as root and type:
   passwd db2inst1
   then type in the new password.
2. Update the DB2 related password in the OpenStack configuration file:
   a. Update the keystone configuration file on Central Server 2:
      1) Find the keystone configuration file:
         cd /etc/keystone
         grep -Fnre connection ./
         ./keystone.conf:28:connection =
         W01CTTp2MV12b3pfcw9FZm46Ly94ZnFv0mZwYjRmaWdAMTcyLjE2LjM0LjIwMTo1MDAwMC9iY3JhZmducA==
b. Decode the DB2 connect information:

```shell
openstack-obfuscate
-u W0lCTTp2MV12b3pcf9fz3m6Ly94znfV0mZwYmdyZmdAMTAlMDYuMzQuMjAxOjUwMDAwL23jcmFmZ25w
```

ibm_db_sa://ksdb:scotest@10.16.34.201:50000/openstack

---

c. Encrypt the DB2 connect information with the new password:

```shell
openstack-obfuscate ibm_db_sa://ksdb:<type here the new password>@10.16.34.201:50000/openstack
```

W0lCTTp2MV12b3pcf9fz3m6Ly94znfV0mZwYmdyZmdAMTAlMDYuMzQuMjAxOjUwMDAwL23jcmFmZ25w

---

d. Update the other OpenStack components on the Region Server.

3. Find the configuration files and replace the DB2 connection information with the new password.

On a KVM region, find the configuration files:

```bash
cd /etc
grep -Fnr sql_connection ./
```

```bash
grep -Fnr sql_connection ./
```

`.glance/glance-registry.conf:27:sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA= ./glance-api.conf:32:sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA= ./cinder/cinder.conf:17:sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA= ./nova/nova.conf:71:sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA=

On a VMware region, find the configuration files:

```bash
cd /etc
grep -Fnr sql_connection ./
```

```bash
grep -Fnr sql_connection ./
```

`.glance/glance-registry.conf:27:sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA= ./glance-api.conf:32:sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA= ./cinder/cinder.conf:17:sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA= ./nova/nova.conf:71:sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA= ./nova/nova.conf:142:smartcloud_sql_connection = W0lCTTp2MV12b3pcf9fz3m6Ly90exi1NDQOZmpcG10zm1qDE3M14xNi4znNC4yMDE6NTAwMDAVYmNzYWZnbnA=

After modifying the configuration files, you must restart all the OpenStack services. To test the changes has been correctly done, execute an OpenStack command. For example, try to deploy an instance running the command `nova boot`.

Change the `db2inst1` password on Workload Deployer:

a. Login as root on Central Server 3.


c. Run the command:

```bash
service iwd setdb2host $DB2_HOST_ADDRESS $DB2_HOST_PORT $DB2_USER $DB2_PASS
```

d. Modify the parameter `DB2_PASS` in the files `/etc/rc.d/init.d/iwd-env` and `/etc/init.d/iwd-env`. Pay attention to the value `DB2_PASS`, it must be an encrypted password for the DB2 user, `dbinst1`. Here is an example on how to get the encrypted password on Central Server 2:

```bash
vim /tmp/tmp_password.txt
```

```
password=$db2_password_in_plaintext
```

then run the command to encrypt the DB2 password:
/IBM/WebSphere/AppServer/bin/PropFilePasswordEncoder.sh /tmp/tmp_password.txt <new password>

Then you can find the encrypted password in /tmp/tmp_password.txt

**Note:** Pay attention to the format when modifying the password in file iwd-env.

**e.** Restart the Workload Deployer by running:

```
service iwd restart
```

The output should look like:

```
[root@CS-3 init.d]# service iwd status
Processes:
    db2sysc : STOPPED
derby : STOPPED
    storehouse : RUNNING ( 2839 )
kernelservices : RUNNING ( 2917 )
zso : RUNNING ( 3398 )
```

### Changing the bpmuser password

To change the password for the DB2 user used by the Business Process Manager (bpmuser) you must change it both on Central Server 1 and in the WebSphere Application Server console used by the Business Process Manager.

1. To change the password for the DB2 user used by the Business Process Manager (bpmuser log on as root on Central Server 1 and launch:

```
passwd bpmuser
```

2. To change the Business Process Manager DB password in WebSphere Application Server, follow these steps:

   a. Log on to the Business Process Manager WebSphere Application Server console: https://$central-server-4:9043/ibm/console/logon.jsp as tw_admin
   b. Select **Resources**.
   c. Select **JDBC**.
   d. Select **Data sources** and click **BPM Business Space data source**.
   e. Click the option **JAAS - J2C authentication data**.
   f. Click **BPM_DB_ALIAS**.
   g. Insert the new password.
   h. Click **Apply** to validate the change.
   i. A message informs you that you must save your change.
   j. Click **Save directly to the master configuration**.
   k. Click **Synchronize**.
   l. Test the DB connection by clicking **Test connection** and selecting **BPM Business Space data source**.

**Note:** If you get errors while synchronizing the changes, try to log out and log in the page, and try to modify the password again.


**Note:** To change the Workload Deployer DB2 connection password, change the DB2 password for the Workload Deployer:
Changing the vCenter password

When you change the vCenter password, run the following steps in your IBM Cloud Orchestrator environment:

1. For nova compute, on the Region Server, perform the following steps:
   a. Run `/usr/bin/openstack-obfuscate <new vCenter password>` to obtain the encrypted password.
   b. On the Region Server, edit `/etc/nova/nova.conf` replacing the old value of `host_password` with the new value generated in the previous step.
   c. Restart the openstack-nova-compute service.

2. For cinder, on the Region Server, perform the following steps:
   a. Run `/usr/bin/openstack-obfuscate <new vCenter password>` to obtain the encrypted password.
   b. Edit `/etc/cinder/cinder.conf` replacing the old value of `vmware_host_password` with the new value generated in the previous step.
   c. Restart the openstack-cinder-volume service.

Changing the password for OpenStack service users

OpenStack has three service users: nova, cinder and glance.

By default, the service users are created with the same password assigned at installation time to the admin user. If you would like to change their passwords follow these procedures:

To change the password for nova user:

1. Change the password in the IBM Cloud Orchestrator user interface by editing the nova user in the Users panel (Administration > Users).
2. On the Region Server run `/usr/bin/openstack-obfuscate <new password for nova user>` to obtain the encrypted password.
3. On the Region Server, do a backup copy of `/etc/nova/nova.conf`.
4. On the Region Server, edit `/etc/nova/nova.conf` replacing the old value of `admin_password` with the newly generated in Step 2.
   **Attention:** It is recommended to comment out the property `admin_token` in nova.conf.
5. Restart all openstack-nova* services.

To change the password for glance user:

1. Change the password in the IBM Cloud Orchestrator user interface by editing the glance user in the Users panel (Administration > Users).
2. On the Region Server run `/usr/bin/openstack-obfuscate <new password for glance user>` to obtain the encrypted password.
3. On the Region Server, do a backup copy of `/etc/glance/glance-api.conf;/etc/glance/glance-registry.conf`.
4. On the Region Server, edit `/etc/glance/glance-api.conf` replacing the old value of `admin_password` with the newly generated in Step 2. Do the same in `/etc/glance/glance-registry.conf`.
   **Attention:** It is recommended to comment out the property `admin_token` in both glance-api.conf and glance-registry.conf.
5. Restart all openstack-glance* services.

To change the password for cinder user:

1. Change the password in the IBM Cloud Orchestrator user interface by editing the cinder user in the Users panel (Administration > Users).
2. On the Region Server run /usr/bin/openstack-obfuscate <new password for cinder user> to obtain the encrypted password.
3. On the Region Server, do a backup copy of /etc/cinder/cinder.conf.
4. On the Region Server, edit /etc/cinder/cinder.conf replacing the old value of admin_password with the newly generated in Step 2.
   **Attention:** It is recommended to comment out the property admin_token in cinder.conf.
5. Restart all openstack-cinder* services.

Starting and stopping the services in the deployment service node

You can manage the services that run in the deployment service node.

This table illustrates how you can manage the services that run in the deployment service node.

**Table 14. Managing deployment service node services**

<table>
<thead>
<tr>
<th>Services</th>
<th>Check status</th>
<th>Start command</th>
<th>Stop command</th>
</tr>
</thead>
<tbody>
<tr>
<td>http service</td>
<td>service httpd status</td>
<td>service httpd start</td>
<td>service httpd stop</td>
</tr>
<tr>
<td>chef server</td>
<td>/opt/chef-server/</td>
<td>/opt/chef-server/</td>
<td>/opt/chef-server/</td>
</tr>
<tr>
<td></td>
<td>bin/chef-server-ctl</td>
<td>bin/chef-server-ctl</td>
<td>bin/chef-server-ctl</td>
</tr>
<tr>
<td></td>
<td>status</td>
<td>start</td>
<td>stop</td>
</tr>
<tr>
<td>heat api</td>
<td>service</td>
<td>service</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-api</td>
<td>openstack-heat-api</td>
<td>openstack-heat-api</td>
</tr>
<tr>
<td></td>
<td>status</td>
<td>start</td>
<td>stop</td>
</tr>
<tr>
<td>heat engine</td>
<td>service</td>
<td>service</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-engine</td>
<td>openstack-heat-engine</td>
<td>openstack-heat-engine</td>
</tr>
<tr>
<td></td>
<td>status</td>
<td>start</td>
<td>stop</td>
</tr>
<tr>
<td>ds api</td>
<td>service</td>
<td>service</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>ds-api status</td>
<td>ds-api start</td>
<td>ds-api stop</td>
</tr>
<tr>
<td>ds engine</td>
<td>service</td>
<td>service</td>
<td>service</td>
</tr>
<tr>
<td></td>
<td>ds-engine status</td>
<td>ds-engine start</td>
<td>ds-engine stop</td>
</tr>
</tbody>
</table>

**Nova**

<table>
<thead>
<tr>
<th>service</th>
<th>openstack-nova-api</th>
<th>openstack-nova-api</th>
<th>openstack-nova-api</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>start</td>
<td>stop</td>
<td>stop</td>
</tr>
<tr>
<td>service</td>
<td>openstack-nova-scheduler</td>
<td>openstack-nova-scheduler</td>
<td>openstack-nova-scheduler</td>
</tr>
<tr>
<td>status</td>
<td>start</td>
<td>stop</td>
<td>stop</td>
</tr>
<tr>
<td>service</td>
<td>openstack-nova-network</td>
<td>openstack-nova-network</td>
<td>openstack-nova-network</td>
</tr>
<tr>
<td>status</td>
<td>start</td>
<td>stop</td>
<td>stop</td>
</tr>
<tr>
<td>service</td>
<td>openstack-nova-conductor</td>
<td>openstack-nova-conductor</td>
<td>openstack-nova-conductor</td>
</tr>
<tr>
<td>status</td>
<td>start</td>
<td>stop</td>
<td>stop</td>
</tr>
<tr>
<td>service</td>
<td>openstack-nova-compute</td>
<td>openstack-nova-compute</td>
<td>openstack-nova-compute</td>
</tr>
<tr>
<td>status</td>
<td>start</td>
<td>stop</td>
<td>stop</td>
</tr>
</tbody>
</table>
### Configuring high availability

Configure your IBM Cloud Orchestrator environment to use high-availability capabilities.

For information about high availability, see "Deployment modes" on page 11, "Deployment topologies" on page 16, and "Managing high availability" on page 187.

For information about installing the high-availability topology, see "Deploying the Distributed Active-Active (High Availability) topology" on page 69.

#### Configuring the external database server for the high-availability topology

The external database must be monitored by System Automation Application Manager because many IBM Cloud Orchestrator processes depend on the external database.

**About this task**

The database instance is a key component of IBM Cloud Orchestrator, and should be installed with high-availability features (for example, DB2 HADR or DB2 with shared disk). For more information, see Configuring a clustered environment using DB2 High Availability Instance Configuration Utility (db2haicu) in the IBM DB2 documentation [http://www-01.ibm.com/support/knowledgecenter/SSEP GG_10.5.0/com.ibm.db2.luw.admin.ha.doc/doc/t0052800.html](http://www-01.ibm.com/support/knowledgecenter/SSEP GG_10.5.0/com.ibm.db2.luw.admin.ha.doc/doc/t0052800.html).

Any high-availability solution should also manage the DB2 nosql service.

The control script provided (db2ctrl) is a sample script and should be adapted to suit your installation. The DB2 database service is configured in System Automation Application Manager as an agentless adapter resource. This configuration enables System Automation Application Manager to connect to a highly available database via the virtual IP address, and monitor and manage DB2 automatically on the active node. If the database is configured as highly available, you must adapt the commands that are used to manage and monitor the database.

During installation, the saamuser user is created automatically on various IBM Cloud Orchestrator servers. However, you must manually create the saamuser user on the external database server. If you have more than one external database server, create the saamuser user on each server in your database cluster. After you
create the saamuser user, you must manually create the control scripts and configure sudo to enable System Automation Application Manager to monitor and manage the external database.

**Tip:** The example commands in the following procedure are for a UNIX server. If the operating system of the external database server is not UNIX, replace the example commands with the equivalent commands.

**Procedure**

1. Log in to the external database server as a root user.
2. Create the saamuser user:
   ```
   # useradd -m saamuser
   ```
3. Create the password for the saamuser user:
   ```
   # passwd saamuser
   New password:
   Retype new password:
   When prompted for the password, enter the password that is used by the saamuser user on the IBM Cloud Orchestrator servers.
   ```
4. Create the `/opt/IBM/tsamp/eez/scripts` directory:
   ```
   # mkdir -p /opt/IBM/tsamp/eez/scripts
   ```
5. Copy the following files from the System Automation Application Manager server to the specified directories on the IBM DB2 server:
   - Copy `/opt/IBM/tsamp/eez/scripts/servicectrl.sh` to `/opt/IBM/tsamp/eez/
   - Copy `/opt/IBM/tsamp/eez/scripts/db2ctrl` to `/etc/init.d`
6. If necessary, edit the `/etc/init.d/db2ctrl` file to suit your IBM DB2 installation.
7. Ensure that the saamuser user is the owner of the `/opt/IBM/tsamp/eez/scripts` directory and its contents:
   ```
   # chown -R saamuser
   ```
8. Add the db2ctrl script as a service:
   ```
   chkconfig --add db2ctrl; chkconfig db2ctrl off
   ```
9. Disable any autostart of the IBM DB2 services for the external database.
10. Configure sudo for the saamuser user:
   a. Create the `/etc/sudoers.d/saam` file with the following content:
      ```
      # sudoers additional file for /etc/sudoers.d/
      # IMPORTANT: This file must have no / or . in its name and file permissions
      # must be set to 440
      # This file is for the IBM System Automation Application Manager 1D to call
      # the IBM DB2 control script that is provided with IBM Cloud Orchestrator
      Defaults::saamuser !requiretty
      # scripts found in control script directory
      Cmd_Alias SACTRL = /sbin/service
      # allow for root access
      saamuser ALL = (root) NOPASSWD: SACTRL
      ```
   b. Change the file permissions of the new file:
      ```
      # chmod 440 /etc/sudoers.d/saam
      ```
   c. Ensure that the `/etc/sudoers` file contains the following line:
      ```
      #includedir /etc/sudoers.d
      ```
11. If you have more than one external database server, repeat this procedure to create the saamuser user on each server in your database cluster.
Results

System Automation Application Manager can now use the saamuser user to monitor the external database.

Enabling autostart of System Automation Application Manager

Because System Automation Application Manager is not started when the virtual machine where it is installed starts, you must enable autostart of System Automation Application Manager at startup.

To automatically start System Automation Application Manager at the startup of the virtual machine where it is installed, complete the following steps:

1. Log on, as a root user, to the virtual machine where System Automation Application Manager is installed.

2. Create a script named saam.conf in the /etc/init directory, and insert the following content:

```bash
#IBM System Automation Application Manager Startup
description 'IBM System Automation Application Manager'
version 'SAAM 4.1.0.0'
start on stopped rc RUNLEVEL=[345]
stop on starting rc RUNLEVEL=[016]
console output
pre-start script
    su db2inst1 -c /home/db2inst1/sqllib/adm/db2start
    sleep 10
    /opt/IBM/WebSphere/AppServer/bin/startServer.sh server1
    sleep 30
    eezdmn start
    eezaladapter start
end script
post-stop script
    eezaladapter stop
    eezdmn -SHUTDOWN
    /opt/IBM/WebSphere/AppServer/bin/stopServer.sh server1 -username wasadmin -password <password>
    su db2inst1 -c /home/db2inst1/sqllib/adm/db2stop
end script

By default, the wasadmin user ID is used to install the WebSphere Application Server automatically in the System Automation Application Manager installation. The password for this user is the password provided in the parameter OrchestratorPassword. You must replace this value in the script.

3. Make sure that the script has the correct security settings by running the following command:

   chmod 644 /etc/init/saam.conf

If you start System Automation Application Manager now, it will fail because the agentless adapter is not correctly configured. You configure the agentless adapter in the next section.

After you configure the agentless adapter, you can start and stop System Automation Application Manager by running the start saam and stop saam commands. System Automation Application Manager is automatically started when the server is started.
Configuring System Automation Application Manager

Configure System Automation Application Manager to work with IBM Cloud Orchestrator.

About this task

Note: Before proceeding, ensure that you have completed the task "Configuring the external database server for the high-availability topology" on page 145.

During the IBM Cloud Orchestrator installation process, all components and configuration artifacts that are needed to manage the complete stack with System Automation Application Manager are installed and prepared. To activate the System Automation Application Manager management solution, complete the following procedure:

Procedure

1. Configure the Agentless Adapter:
   a. During the installation of the Central Servers and Region Servers, all necessary automation policies are automatically created on the node where System Automation Application Manager is installed. These policies are stored in the /etc/opt/IBM/tsamp/eez/aladapterPolicyPool directory.
   b. Identify the domain names for all Agentless Adapter policy XML files that you want to manage with System Automation Application Manager. The file for the Central Servers is named SCOcentralALA.xml, and the file for each Region Server is named <region_name>.xml. The domain name is enclosed within the <AutomationDomainName> tag. The /etc/opt/IBM/tsamp/eez/aladapterPolicyPool directory also contains some sample xml files and an xsd file; ignore these files. You can use the following grep command to identify the domain names to be managed:

   ```bash
grep -h "<AutomationDomainName>" `ls /etc/opt/IBM/tsamp/eez/aladapterPolicyPool/*.xml | grep -v Sample` | sort | uniq | cut -d » -f 2 | cut -d |
   ```

   The output of this command is a list of domain names, as shown in the following example:

   ```bash
   RegionKVMCompute
   RegionKVMNeutron
   RegionVMwareNeutron
   SCOcentralALA
   ```

   These domains are managed by the agentless adapter, as follows:
   - You must have one domain, SCOcentralALA, for the central servers.
   - Each KVM Region has two agentless adaptor domains: <region-name>Compute for the compute nodes of this region, and <region name>Neutron for the Neutron server of the region.
   - Each VMware region has only one agentless adapter domain: <region name>Neutron for the Neutron server of this region.

   These regions are managed by the agentless adapter and require a nonroot SSH access. To provide this nonroot access, the user saamuser is installed on all servers, with the password specified in the OrchestratorPassword parameter. You use this user name and password in later steps.
   c. Use the cfgeezdmn command to generate one Agentless Adapter domain for every domain name identified in the previous step. The cfgeezdmn command requires an X11 display. Use VNC, X11 forwarding, local X11 console, or any other method to provide a valid X11 display before you run
the command. For more information, see Configuring Agentless Adapters in the System Automation Application Manager, Installation and Configuration Guide.

To configure the automation domain names identified in step b, click the Non-clustered Nodes tab, and select Enable local Agentless Adapter configuration. If you are doing this step for the first time, the configuration page for the agentless adapter opens automatically. Later, you can open the configuration page by clicking Configure. On this page, add all domain names by clicking Add and then click OK.

d. Click the User Credentials tab to enter the user IDs and passwords that are used by the agentless adapter to communicate with the nodes. You can either enter the generic saamuser user ID with the same password for all nodes with agentless adapter resource, or specify the saamuser user ID and password combination for each individual servers. During the installation of IBM Cloud Orchestrator, the saamuser user is automatically created for nonroot access, with the password specified in the OrchestratorPassword parameter. This user ID and password are used to manage all resources managed by the Agentless Adapter. To save these settings, click Save and then click OK.

2. The agentless adapter is now configured. System Automation Application Manager can now be started. Without a valid configuration, the agentless adapter start fails. Restart the System Automation Application Manager by issuing the following commands: stop saam; start saam; stop saam; start saam. After the agentless adapter is configured, System Automation Application Manager starts correctly.

3. Configure the System Automation Application Manager Automation Engine. Use the cfgeezdmn command, as described in Configuring System Automation Application Manager in the System Automation Application Manager, Installation and Configuration Guide.

Click the Application Manager tab and click Configure. In the new page, click the User Credentials tab. The user credentials are the user IDs and passwords used by the Automation Engine to access the adapter domains: either an agentless adapter domain or a System Automation for Multiplatforms adapter domain. Different types of domain are managed differently and require different credentials for access. The names of the agentless adapter domains are the same as used in previous steps, and you must use the same combination of saamuser and password as earlier. The agentless adapter runs locally on the System Automation Application Manager virtual machine. The agentless adapter uses nonroot SSH to connect to the virtual machine where the resources are running. IBM Cloud Orchestrator V2.4 supports nonroot SSH, and installs a default user saamuser on all systems managed as agentless adapter domain. Therefore, you must enter saamuser and the password of saamuser on the System Automation Application Manager virtual machine for the agentless adapter domains. The adapter of System Automation for Multiplatforms domains does not run on the System Automation Application Manager virtual machine, but on the servers that are managed by the System Automation for Multiplatforms domains. These servers are accessed by the automation engine, but not through SSH, and do not require root SSH access. However, System Automation for Multiplatforms is installed as root user, so you must use the root user name and password on the virtual machines where the System Automation for Multiplatforms domain is running. The virtual machines of one System Automation for Multiplatforms must have the same root password.

Run the following command to display a list of all domains:

grep "<AutomationDomain>" /etc/opt/IBM/tsamp/eez/policyPool/SCOsaam.xml | sort | uniq | cut -d ">" -f 2 | cut -d "<" -f 1

Chapter 2. Installing 149
The output of this command is a list of domain names, as shown in the following example:

cs2Domain
RegionKVM
RegionKVMCompute
RegionKVMNeutron
RegionVMware
RegionVMwareNeutron
SCOcentralALA

This list includes all the agentless adapter domains from step 1. It also contains the System Automation for Multiplatforms domains. The System Automation for Multiplatforms domain that ensures the high availability of the Central Server 2 cluster is named cs2Domain. This cluster is running on the Central Server 2 primary and secondary virtual machines. Each Region has a System Automation for Multiplatforms domain, which has the same name as the Region. This cluster runs on the Region Server primary and secondary nodes. For all of these domains, click Add... Enter the domain name, user ID, password, and password confirmation fields and click OK. Repeat this step for each domain that is listed in the output of the grep command above. Use the credentials as explained above. Click Save and OK. To exit from cfgeezdmn, click Done.

Start cfgeezdmn again and switch again to the user credentials tab of the application manager configuration. Click a domain to select and highlight it. Now you can click Validate to check the user credential settings for this domain. Click the Command Shell tab. In User authentication for invoking the end-to-end automation manager shell, select Use specified user credentials and enter the eezadmin userid and the password specified in the OrchestratorPassword parameter. In User authentication for issuing commands against first-level automation (FLA) domains, select Use FLA domain access credentials as defined under User credentials. Finally, save your changes by clicking Save and OK. Close cfgeezdmn by clicking Done. You have completed the configuration of the automation engine, which can now access the adapters.

Restart System Automation Application Manager by issuing stop saam; start saam.

4. Verify that the domains are set up correctly:

eeecsc -c "lseezdom"

5. Log on to DASH, which is the web user interface of System Automation Application Manager: https://<hostname of saam>:16311/ibm/console with the user eezadmin and its password.

6. Enter the credentials for the DASH to access the automation engine domains. To do this, click the Explore Automation Domains tab. If the Domain Health State shows Not logged in, right-click the domain and click Log in. These are the credentials that the DASH GUI uses to connect to the adapters. Enter the same user IDs and passwords that you used for the credentials in step 3 for the automation configuration. The UI shows the same domains as in step 3. If you want to store the logon information for this UI session, select Save in credential store. For more information, see Activating an automation policy in the System Automation Application Manager, Administrator's and User's Guide.

7. You can now see the state of the System Automation for Multiplatforms domains. For the domains managed by the agentless adapter, you also have to active the policy for these domains. To do this, right-click the domain and select Open Policy Activation Page. This opens a new tab. Select the domain
and, in the right part of the screen, a list of all policies available for this agentless automation domain is displayed. By default, there is only one. Right-click it and select **Activate Policy**.

8. Activate the End-To-End automation policy named `SCOsaam.xml`. If the end-to-end automation policy is activated, the automation engine enforces the policy rules and starts all services. To activate, right-click the `SCOsaam` policy and select **Open Policy Activation Page**. This opens a new tab. Select the domain and, in the right part of the screen, a list of all policies available for this agentless automation domain is displayed. By default, there is only one. Right-click it and select **Activate Policy**. For more information, read **Activating an automation policy** in the System Automation Application Manager, Administrator's and User's Guide.

9. To manage the end-to-end automation, switch to **Operate end-to-end resources**.

**Results**

System Automation Application Manager is configured and the policies are activated. System Automation Application Manager checks the status of the IBM Cloud Orchestrator management services and starts them if needed. After the services are started, System Automation Application Manager detects any outage of the IBM Cloud Orchestrator management services and automatically restarts them.

*If you want to stop any services, see “Controlling the management stack” on page 195.*

**Post-installation configurations**

For certain features, you must perform specific configurations on the hypervisor managers for IBM Cloud Orchestrator.

**Supporting VMware vSphere 5.0**

Users of vSphere 5.0 must host their WSDL files locally. These steps are applicable for vCenter 5.0 or ESXi 5.0 and you can either mirror the WSDL from the vCenter or ESXi server that you want to use, or you can download the SDK directly from VMware. These workaround steps fix a known issue with the WSDL that was resolved in later versions. When setting the VMware VCDriver configuration options, you must include the `wsdl_location` option. For more information, see the OpenStack documentation at: [http://docs.openstack.org/trunk/config-reference/content/vmware.html#VMWare_additional_config](http://docs.openstack.org/trunk/config-reference/content/vmware.html#VMWare_additional_config)

**Supporting Cinder volumes in VMware regions**

To correctly exploit the creation and the management of block storages, and optimize their placement in VMware regions, you must define storage policies in vCenter. For more information, see the OpenStack documentation at [http://docs.openstack.org/trunk/config-reference/content/vmware-vmdk-driver.html](http://docs.openstack.org/trunk/config-reference/content/vmware-vmdk-driver.html)
Upgrading from SmartCloud Orchestrator V2.3 or V2.3.0.1

You can upgrade from SmartCloud Orchestrator V2.3 or V2.3.0.1 to IBM Cloud Orchestrator by using the ico-upgrade-tool.

Before you begin

Make sure that:

- The last SmartCloud Orchestrator version is V2.3 or V2.3.0.1.
- SmartCloud Orchestrator V2.3 or V2.3.0.1 hardware fulfills the IBM Cloud Orchestrator V2.4 hardware prerequisites described in "Hardware prerequisites" on page 19.
- The SmartCloud Orchestrator V2.3 or V2.3.0.1 servers must run Red Hat Enterprise Linux 6.4 or 6.5. Make sure that your SmartCloud Orchestrator V2.3 or V2.3.0.1 Central Servers, Region Servers, and compute nodes (for a KVM region) are running on Red Hat Enterprise Linux 6.4 or 6.5. For more information, see "Software prerequisites" on page 21. If you need to upgrade Red Hat Enterprise Linux, reboot your servers after the upgrade procedure completes.
- The IBM Cloud Orchestrator Deployment Service is installed and can connect to all IBM SmartCloud Orchestrator V2.3 Central Servers, Region Servers, and compute nodes (for a KVM region). The Red Hat Enterprise Linux version of the Deployment Server must be the same as the Red Hat Enterprise Linux version of the SmartCloud Orchestrator V2.3 or V2.3.0.1 servers.
- There are three kinds of DNS options in SmartCloud Orchestrator V2.3. If option A (built-in DNS) or option B (built-in DNS with corporate DNS as parent) is used, edit the /etc/resolv.conf file on the deployment-service server and add a nameserver entry that points to the Central Server 1. If option C (corporate DNS) is used, edit the /etc/resolv.conf file on the deployment-service server and add a nameserver entry that points to the corporate dns server.
- If you use RHEL 6.5 as the operating system to host SmartCloud Orchestrator V2.3 central servers, before upgrading to IBM Cloud Orchestrator V2.4, check the version of openssl package on Central Server 3 and upgrade it at least to version openssl-1.0.1e-16.el6.x86_64 to avoid the upgrade procedure from failing when mapping images between the Workload Deployer component and OpenStack.
- For VMmware Region Server upgrade, users of vSphere 5.0 or earlier must host their WSDL files locally. These steps are applicable for vCenter 5.0 or ESXi 5.0 and you can either mirror the WSDL from the vCenter or ESXi server that you intend to use or you can download the SDK directly from VMware. Refer to section vSphere 5.0 and earlier additional set up and Procedure 2.1. To mirror WSDL from vCenter (or ESXi) in OpenStack, see the OpenStack VMware vSphere configuration for how to prepare WSDL files and configure wsdl_location.

Note: Make sure that the VMware region nova user has permission to access the folder and file configured for wsdl_location in the /etc/nova.conf file. For upgrade from SmartCloud Orchestrator V2.3, you do not need to configure wsdl_location in the VMware region's /etc/nova.conf file. Instead, you must configure the vmware-wsdlloc option in the upgrade configuration file. For more information, see "Configuring the upgrade configuration file" on page 159.

Note: By default, the first time the ico-upgrade-tool discover command is called (it needs SmartCloud Orchestrator 2.3 started), it will connect to the
SmartCloud Orchestrator 2.3 servers and collect information, and the information will be saved to the Deployment Service server. Later, a user calls the ico-upgrade-tool discover command without the --force option (you do not need SmartCloud Orchestrator 2.3 started), it will just read the saved information, to force this command connect to the SmartCloud Orchestrator 2.3 server and collect information again, run this command with the --force option (you must have SmartCloud Orchestrator 2.3 started whenever you run discover command with --force option):

```
ico-upgrade-tool discover -C YOUR_SC023_CS1_ADDR -p PASSWORD --admin-user SC023_ADMIN_USER --admin-pass SC023_ADMIN_PASSWORD --force -f UPGRADE_CLIP
```

So in general, you must make sure SmartCloud Orchestrator 2.3 servers are started in the following 2 scenario:

1. When you run ico-upgrade-tool discover for the first time for SmartCloud Orchestrator 2.3 on your Deployment Server.
2. When you run ico-upgrade-tool discover with the --force option.

Both scenarios will try to connect SmartCloud Orchestrator 2.3 to collect OpenStack information like endpoints, hypervisor lists, and call IBM Workload Deployer commands to collect Workload Deployer images mappings and so on.

- If you have IBM SmartCloud Orchestrator Content Pack for UrbanCode Deploy installed in SmartCloud Orchestrator V2.3, you must install the new version from IBM Cloud Orchestrator before upgrading from SmartCloud Orchestrator V2.3 to IBM Cloud Orchestrator V2.4.

**Restriction:** The self-service categories **Virtual System Operations (Single Instance)** and **Virtual System Operations (Multiple Instances)** are not migrated. These categories and related offerings are replaced with new offerings in IBM Cloud Orchestrator V2.4. If you added your own customized offerings to either of these categories and you want to migrate your offerings, create a new self-service category, and move your offerings from **Virtual System Operations (Single Instance)** and **Virtual System Operations (Multiple Instances)** to your new self-service category. You can move a self-service offering between categories by editing the offering and changing the related category in the offering details.

**About this task**

The migration tool performs the following steps:

1. Checks the upgrade prerequisites.
2. Collects the SmartCloud Orchestrator V2.3 or V2.3.0.1 installation information which is used for the upgrade.
3. Backs up the OpenStack configuration files. All configuration files are backed up to /opt/ibm/openstack_backup/sco23_upgrade.
4. Upgrades DB2 on the Central Servers and the Region Servers from V10.1 to V10.5.
5. Upgrades the IBM Cloud Orchestrator components to version 2.4, and performs the database migration for the VMware regions.

After the upgrade, all the services remain on the same server as in SmartCloud Orchestrator V2.3 or V2.3.0.1, with the following exceptions:

- Virtual Image Library and the iaasgateway service are stopped on Central Server 2.
- SmartCloud Orchestrator user interface is moved from Central Server 3 to Central Server 4.
SmartCloud Entry service is removed from VMware region and replaced with the new nova service.

ico-upgrade-tool is run on the Deployment Service server of IBM Cloud Orchestrator V2.4. It has the following subcommands:


where <subcommand> is one of the following subcommands:

discover
   Generates the configuration file for the upgrade.

list
   Lists the topology of the last installed SmartCloud Orchestrator release.

upgrade
   Upgrades SmartCloud Orchestrator V2.3 to IBM Cloud Orchestrator V2.4.

help
   Displays help about this program or one of its subcommand.

Use the ico-upgrade-tool help subcommand to get detailed usage of the subcommands. For example, to get information about the upgrade subcommand, run the following command:

ico-upgrade-tool help upgrade

Usage:

ico-upgrade-tool upgrade -C <central server 1 IP address>
   [--port <central server 1 ssh port>]
   [-u <central server 1 user name>]
   [-p <central server 1 user password>]
   [--key-file <central server 1 private ssh-key>]

Upgrade the SmartCloud Orchestrator V2.3 or V2.3.0.1 to IBM Cloud Orchestrator V2.4.

Optional arguments:

-C <Central Server 1 IP address>, -- central-srv 1 < Central Server 1 IP address>
   The IP address of Central Server 1.
--port <Central Server 1 ssh port>
   The ssh port of Central Server 1.
-u <Central Server 1 user name>, --user <Central Server 1 user name>
   The user name on Central Server 1.
-p <Central Server 1 user password>, -- password <Central Server 1 user password>
   The password of the user on Central Server 1.
--key-file <Central Server 1 private ssh-key>
   The private ssh-key on Central Server 1.

To upgrade SmartCloud Orchestrator V2.3 to IBM Cloud Orchestrator V2.4, perform the following steps:

Procedure

1. Back up the Central Server and Region Server virtual machines and the Compute Nodes.
   
   For VMware-hosted Central and Region Servers, see Taking Snapshots (VMware) to take snapshots for Central Servers and Region Servers.

2. Source /root/openrc on the Deployment Server and use the ico-upgrade-tool discover command to discover the SmartCloud Orchestrator topology, and generate the upgrade configuration file to be used by the upgrade procedure.
ico-upgrade-tool discover -C CENTRAL_SRV1_IP -p CENTRAL_SRV1_ROOT_PASS
    --admin-user SCO23_ADMIN_USER
    --admin-pass SCO23_ADMIN_PASSWORD
    -f UPGRADE_FILE_LOCATION

where:

**CENTRAL_SRV1_IP**
- Specifies the IP address of Central Server 1.

**CENTRAL_SRV1_ROOT_PASS**
- Specifies the root password of Central Server 1.

**SCO23_ADMIN_USER**
- Specifies the user name of the SmartCloud Orchestrator V2.3 admin user.

**SCO23_ADMIN_PASSWORD**
- Specifies the user password of the SmartCloud Orchestrator V2.3 admin user.

**UPGRADE_FILE_LOCATION**
- Specifies the location where the upgrade configuration file is stored.

**vmdatastore**
- The VMware datastore name. It will set option datastore_regex to /etc/nova/nova.conf, which can be a regular expression. For example, if you have datastore001, datastore002, datastore003, you can set vmdatastore to datastore00* to match those datastores.

**Note:** datastore_cluster_name does not support regular expression. Only one cluster can be set to this option, this is the current design.

**Note:** If SmartCloud Orchestrator V2.3 uses an external database, run the ico-upgrade-tool discover command with the --extdb option:

    ico-upgrade-tool discover -C CENTRAL_SRV1_IP -p CENTRAL_SRV1_ROOT_PASS
    --admin-user SCO23_ADMIN_USER
    --admin-pass SCO23_ADMIN_PASSWORD
    --extdb -f UPGRADE_FILE_LOCATION

If you run the ico-upgrade-tool discover command again, it just reads the saved information. In this case, you do not need to have SmartCloud Orchestrator started. To force this command, connect to the SmartCloud Orchestrator V2.3 server and collect the information again. Run this command with the --force option.

    ico-upgrade-tool discover -C YOUR_SCO23_CS1_ADDR -p PASSWORD
    --admin-user SCO23_ADMIN_USER --admin-pass SCO23_ADMIN_PASSWORD --force -f UPGRADE_CFG_LOCATION

**Note:** Be sure that the SmartCloud Orchestrator servers are started when you run the ico-upgrade-tool discover for the first time or when you specify the --force option.

You can also discover the SmartCloud Orchestrator topology with the ds wizard, by running ds wizard and selecting

[2] Discover an IBM SmartCloud Orchestrator 2.3.x topology.

3. If all SmartCloud Orchestrator V2.3 or V2.3.0.1 servers are running on Red Hat Enterprise Linux 6.4 or 6.5, go to step 5 directly; otherwise, upgrade all SmartCloud Orchestrator V2.3 or V2.3.0.1 servers to Red Hat Enterprise Linux 6.4 or 6.5. Make sure that your SmartCloud Orchestrator V2.3 or V2.3.0.1 Central Servers, Region Servers, and compute nodes (if you have a KVM region) are running on Red Hat Enterprise Linux 6.4 or 6.5. If not, upgrade
your Red Hat to Red Hat Enterprise Linux 6.4 or 6.5. For more information, see https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/ch-yum.html#s1-yum-upgrade-system. After Red Hat server is upgraded, remove from /etc/fstab the auto mount item for the old Red Hat ISO file, for example, if SmartCloud Orchestrator V2.3 servers are running on Red Hat Enterprise Linux 6.3, remove following items from /etc/fstab:

```
/opt/RHEL6.3-20120613.2-Server-x86_64-DVD1.iso /data/repos/rhel6 iso9660 loop 0 0
```

4. Reboot your servers after the Red Hat system upgrade. If you have upgraded your Red Hat system according to step 3, reboot your servers to make the new Linux kernel work.

5. See “Configuring the upgrade configuration file” on page 159 for information about how to configure the upgrade configuration file.

6. Stop the SmartCloud Orchestrator processes.

If System Automation Application Manager is not used to control SmartCloud Orchestrator, see http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_start_stop_sco.html to start or stop SmartCloud Orchestrator.

If System Automation Application Manager is used to control SmartCloud Orchestrator, see http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_control_mgmt_stack.html to request offline SmartCloud Orchestrator from System Automation Application Manager.

7. Source /root/openrc on the Deployment Server and run the following commands to upgrade SmartCloud Orchestrator V2.3 or V2.3.0.1 to IBM Cloud Orchestrator V2.4:

```
ico-upgrade-tool upgrade -C CENTRAL_SRV1_IP -p CENTRAL_SRV1_ROOT_PASS
```

where:

**CENTRAL_SRV1_IP**

Defines the IP address of Central Server 1.

**CENTRAL_SRV1_ROOT_PASS**

Defines the root password of Central Server 1.

You can also upgrade the SmartCloud Orchestrator V2.3 topology with the ds wizard, by running ds wizard and selecting [3] Upgrade an IBM SmartCloud Orchestrator 2.3.x deployment.

8. For VMware region upgrade, if you are using a datastore cluster or resource pool on your VMware region managed to vCenter, see “Advanced post configuration after a VMware region is upgraded” on page 169.

9. The ico-upgrade-tool command creates deployment service jobs for the upgrade. Wait until the status of all the upgrade jobs is FINISHED.

After the upgrade, the SmartCloud Orchestrator V2.3 OpenStack configuration files are overwritten with the IBM Cloud Orchestrator V2.4 configurations. The customized configuration in SmartCloud Orchestrator V2.3, for example, the LDAP configuration, must be merged back manually. All the SmartCloud Orchestrator V2.3 OpenStack configuration files are backed up to /opt/ibm/openstack_backup/sco23_upgrade/. For example, the keystone configuration files are backed up to /opt/ibm/openstack_backup/sco23_upgrade/keystone on Central Server 2.
After the configuration files are merged back, the corresponding service must be restarted. For example, after the LDAP configuration files are merged back, run the `/etc/init.d/openstack-keystone restart` command to restart the keystone service.

10. If System Automation Application Manager is used to control IBM Cloud Orchestrator, reconfigure System Automation Application Manager by following the procedure described in [http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_conf_saappman.html](http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_conf_saappman.html). Re-configure service autostart on the Central Servers, Region Servers, and Compute Nodes, and re-execute the step to copy control scripts from `/iaas/schorchestrator` to `/home/saam` on all the servers.

**What to do next**

The `ico-upgrade-tool` exits when the upgrade jobs for Central Servers and Region Servers are executed. The tool does not report failure or success status of the job. To check if the upgrade for SmartCloud Orchestrator V2.3 has completed, use the following commands on the Deployment Server:

- `source /root/openrc`
- `ds job-list`

You can see the status of Central Servers upgrade from job `CentralServer_CS1_SUFFIX_Upgrade` and the status of Region Servers upgrade from job `YOUR_REGION_NAME_CS1_SUFFIX_Upgrade`, for example, if your region name is `VmwareRegion`, Central Server IP is `172.17.5.192`, then your job name is `VmwareRegion_172_17_5_192_Upgrade`.

You can see the status of central servers upgrade from job `CentralServer_CS1_SUFFIX_Upgrade` and the status of Region Servers upgrade from job `YOUR_REGION_NAME_CS1_SUFFIX_Upgrade`, for example, if your region name is `VmwareRegion`, the Central Server IP is `172.17.5.192`, then your job name is `VmwareRegion_172_17_5_192_Upgrade`.

To see the job details, use the following command:

- `ds job-show JOB_ID`

where `JOB_ID` is the job ID of the Central Server or Region Server upgrade job.

If the upgrade finished successfully, the status of all the upgrade jobs is `FINISHED`. If an upgrade job failed with status `ERROR`, refer to the `var/log/ds/ds-engine.log` file on the Deployment Server for details.

You can also check the SmartCloud Orchestrator V2.3 upgrade job status with the `ds wizard` by running `ds wizard` and select “[4] Deployment job(s) status.”

If the upgrade failed, you can roll back your IBM Cloud Orchestrator environment with the data backed up.


For KVM hosted Central and Region Servers, to restore Central Servers and Region Servers, see [http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_backup_services.html](http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_backup_services.html).
Note:

- When you run the nova-manage service list command after the VMware region is upgraded, the nova-smartcloud and nova-compute services are marked as XXX since these services are not used in IBM Cloud Orchestrator V2.4. For example:

  # nova-manage service list

<table>
<thead>
<tr>
<th>Binary</th>
<th>Host</th>
<th>Zone</th>
<th>Status</th>
<th>State</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova-scheduler</td>
<td>scotest-d25-region1</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2014-07-30 08:23:50.454159</td>
</tr>
<tr>
<td>nova-conductor</td>
<td>scotest-d25-region1</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2014-07-30 08:23:49.301436</td>
</tr>
<tr>
<td>nova-consoleauth</td>
<td>scotest-d25-region1</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2014-07-30 08:23:45.223702</td>
</tr>
<tr>
<td>nova-network</td>
<td>scotest-d25-region1</td>
<td>internal</td>
<td>enabled</td>
<td>XXX</td>
<td>2014-07-30 08:23:53.312089</td>
</tr>
<tr>
<td>nova-cert</td>
<td>scotest-d25-region1</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2014-07-30 08:23:54.299339</td>
</tr>
<tr>
<td>nova-smartcloud</td>
<td>scotest-d25-region1</td>
<td>internal</td>
<td>enabled</td>
<td>XXX</td>
<td>2014-07-29 18:28:30.055545</td>
</tr>
<tr>
<td>nova-compute</td>
<td>Cluster10172.19.4.11-443</td>
<td>Cluster10172.19.4.11-443</td>
<td>enabled</td>
<td>:-)</td>
<td>2014-07-30 08:23:52.964374</td>
</tr>
<tr>
<td>nova-network</td>
<td>Cluster10172.19.4.11-443</td>
<td>internal</td>
<td>enabled</td>
<td>:-)</td>
<td>2014-07-30 08:23:52.964374</td>
</tr>
</tbody>
</table>

Also you could see those services are XXX when running the "nova availability-zone-list command:

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>internal</td>
<td>available</td>
</tr>
<tr>
<td>- scotest-d25-region1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- nova-conductor</td>
</tr>
<tr>
<td></td>
<td>- nova-scheduler</td>
</tr>
<tr>
<td></td>
<td>- nova-smartcloud</td>
</tr>
<tr>
<td>Cluster10172.19.4.11-443</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- nova-network</td>
</tr>
<tr>
<td>Cluster10172.19.4.11-443</td>
<td>available</td>
</tr>
<tr>
<td></td>
<td>- Cluster10172.19.4.11-443</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- scotest-d25-region1</td>
</tr>
</tbody>
</table>

- For VMware region upgrade, the compute and network services are named openstack-nova-compute-CLUSTER_NAME and openstack-nova-network-CLUSTER_NAME. The openstack-smartcloud, openstack-nova-compute, and openstack-nova-network services are stopped and disabled after the VMware region upgrade. Do not start them on the VMware region servers. You must start only the compute and network services for the related cluster, like openstack-nova-compute-CLUSTER_NAME and openstack-nova-network-CLUSTER_NAME.

Upgrading with an external database

You can upgrade from SmartCloud Orchestrator V2.3 or V2.3.0.1 with an external database to IBM Cloud Orchestrator V2.4. You must manually upgrade the external database to DB2 V10.5 and manually configure the VMware region if it is installed.

Procedure

2. There are new OpenStack components added to IBM Cloud Orchestrator V2.4.
For Central Servers, OpenStack horizon or ceilometer newly added to Central Server 4 in IBM Cloud Orchestrator V2.4, perform the following steps on the database server:

a. Create a DB2 database for OpenStack horizon and ceilometer or use the same DB2 database as in the other OpenStack components.
b. Create a DB2 user for OpenStack horizon and ceilometer.

For each Region Server, OpenStack heat or ceilometer newly added to Region Servers in IBM Cloud Orchestrator V2.4, perform the following steps on the database server:

a. Create a DB2 database for OpenStack heat and ceilometer or use the same db2 database as in the other OpenStack components.
b. Create a DB2 user for OpenStack heat and ceilometer.

3. Upgrade SmartCloud Orchestrator V2.3 to IBM Cloud Orchestrator V2.4 with the command ico-upgrade-tool. For information about upgrading, see “Upgrading from SmartCloud Orchestrator V2.3 or V2.3.0.1” on page 152.

4. The KVM region is upgraded automatically after step 2. For the VMware region, manually migrate the VMware Region database after step 2 is completed. For more information, see “Migrating data in a VMware region” on page 165.

### Configuring the upgrade configuration file

The ico-upgrade-tool collects information from the SmartCloud Orchestrator V2.3 environment and saves it to the upgrade configuration file you specify when you run the ico-upgrade-tool discover command.

#### About this task

The configuration file is saved in JSON format. Make sure that after you edit this file, it can still be parsed as a valid JSON file.

Configurations are generated for Central Servers with the item named as centralserver, and for each region server with the item named as region name, for example, centralserver part is for all central servers, KVMRegion part is for the KVM region, and VMwareRegion part is for the VMware region.

Most of the options are generated with the values collected from SmartCloud Orchestrator V2.3 except what you must do in the following procedure.

#### Procedure

1. Make sure that no option values in the upgrade configuration file are empty.

   **Note:** If your SmartCloud Orchestrator V2.3 environment uses corporate DNS, the SingleSignOnDomain entry in the upgrade configuration file is empty. You must set the SingleSignOnDomain parameter value to the DNS domain name where the manage-from components are installed. For more information about the SingleSignOnDomain parameter, see “Customizing deployment parameters” on page 41.

   **Note:** If your VMware region servers are connected to vCenter later than 5.0, the vmwisdloc option must be empty.

2. For VMware region server, the following parameters are mandatory for the vCenter configuration:
VMwareRegion: {
... 
   "vcenter": {
      ...
      "password": "", # Vcenter user's password
      "user": "", # Vcenter user's name
      "vminterface": "" # Physical Adapter on ESXi server (such as vmnic0 or vmnic1..)
   }
   "vmdataset": "" # VMware datastore name
... 
}

The vmdataset option sets the datastore_regex option in the /etc/nova/nova.conf file. This option can be a regular expression. For example, if you have datastore001, datastore002, and datastore003, you can set vmdataset to datastore00.* to match those datastores.

When your VMware region servers are connected to vCenter 5.0, besides the above mandatory parameters, vmwsdlloc is also mandatory. Follow the steps here to configure vmwsdlloc:

a. Refer to [http://docs.openstack.org/trunk/config-reference/content/vmware.html](http://docs.openstack.org/trunk/config-reference/content/vmware.html) for how to prepare WSDL files and configure option wsdl_location.

b. Make sure region server's nova user has access to the WSDL files on either VMware region server itself or a http server.

c. For SmartCloud Orchestrator 2.3 upgrade, we do not need to configure the VMware region's /etc/nova.conf directly, instead, we should configure the vmwsdlloc option in upgrade configuration file for VMware region servers.

d. Make sure the vmwsdlloc value starts with file:// if the WSDL files are stored on VMware region server locally. If the WSDL files are stored on a http server, the vmwsdlloc value should start with http://.

Make sure that the vCenter user name and password are correct. The vCenter credentials can be validated in either of the following ways:

a. Login to the vSphere Web Client with the user and password, check whether the credential can be authorized successfully.

b. In SmartCloud Orchestrator 2.3, there is a method to validate vCenter credentials from the VMware region. For details refer to the documentation at: [http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_config_sco_vmware.html?lang=en](http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/t_config_sco_vmware.html?lang=en). For example:

   ```
   # /opt/ibm/openstack/iaas/smartscloud/bin/nova-cloud-pingtest 172.19.4.11 admin passw0rd VMware
   { "cloud": { "driver": "", "name": "", "description": "OpenStack managed VMware", "hostname": "172.19.4.11", \n   "username": "scoadmin10", "password": "object00X", "type": "VMware", "port": 443 } }
   {"results": {"status": "Cloud is OK and ready to deploy instances.", "id": "OK", "label": "OK"}}
   # /opt/ibm/openstack/iaas/smartscloud/bin/nova-cloud-pingtest 172.19.4.11 admin wrongpass VMware
   { "cloud": { "driver": "", "name": "", "description": "OpenStack managed VMware", "hostname": "172.19.4.11", \n   "username": "scoadmin10", "password": "object00X", "type": "VMware", "port": 443 } }
   {"results": {"status": "Unreachable Cloud :Login Failed. The user and password combination is invalid.", "id": "UNREACHABLE", "label": "UN"}}
   ```

3. For the KVM region server, the compute node's password is mandatory. All compute nodes for the KVM region are under the computenodes item. The item name for compute nodes are named as the compute node's host name, for example, for a KVM region KVMRegion with compute node scotest-d20-compute, the configuration file looks like this:
4. For the options with default values, only modify them if they have been manually changed after SmartCloud Orchestrator V2.3 was installed. These options must be modified with the new values. For example, if the nova database user's password has been changed after the SmartCloud Orchestrator V2.3 installation, you must modify it with the new password. All plain text passwords in the upgrade configuration file are encrypted after you run the ico-upgrade-tool upgrade command.

5. **External database configuration**

   If an external database is used for SmartCloud Orchestrator V2.3, for each central and region service, an addition option for database IP address is created. There is no default value for database configuration options, for example:
   
   ```json
   "image": {
     "db": {
       "name": "openstack",
       "user": "noa5473",
       "address": "",
       "servicepassword": "passw0rd"
     }
   }
   
   All options with no default value must be input with a valid value.

**Example**

**Sample upgrade configuration file**

Here is an example of a sample upgrade configuration file. The comments after # are not included in the configuration file:

```json
KVMRegion: {
        ....
        "computenodes": {  # All computes for kvm region are put under this item
          "scotest-d20-compute": {  # compute node's hostname
            "password": "passw0rd"
          }
        }
        ....
      }

  "compute": {  # nova configuration
    "db": {  # nova db name
      "name": "openstack",
      "user": "noa5473",
      "servicepassword": "passw0rd"
    },
    "servicepassword": "passw0rd"  # nova service user's password
  },
  "computenodes": {  # KVMRegion's compute nodes list
    "scotest-d20-compute": {  # compute node hostname
      "password": "passw0rd"
    }
  },
  "image": {  # glance configuration
    "db": {
      "name": "openstack",
      "user": "gle5473"
    }
  }
}
```

Chapter 2. Installing   161
"db": {
  "name": "openstac",
  "user": "het20918"
},
"servicepassword": "passw0rd"
},
"osdbpassword": "passw0rd",
"osdbport": "50000",
"regionname": "VMWareRegion",
"smartcloud": {
  "db": {
    "password": "passw0rd",
    "user": "sce20918"
  },
  "vcenter": {
    "clusters": "Cluster1",
    "host": "172.19.4.11",
    "password": "",
    "user": "",
    "vminterface": ""
  }
},
"smartcloudpassword": "passw0rd",
"template": "vmware_region_upgrade_db2",
"volume": {
  "db": {
    "name": "openstac",
    "user": "cir20918"
  },
  "servicepassword": "passw0rd"
}
},
"centralserver": {
  "BrowserSimpleTokenSecret": "rad/M4P8/MIVFGJewQagLw==",
  "SimpleTokenSecret": "8oG7vheB1vwyDgxPul7uzw==",
  "bpm": {
    "db": {
      "name": "BPMDB",
      "password": "passw0rd",
      "port": "50000",
      "user": "bpmuser"
    }
  },
  "cmn": {
    "db": {
      "name": "CMNDB",
      "password": "passw0rd",
      "port": "50000",
      "user": "bpmuser"
    }
  },
  "dashboard": {
    "db": {
      "name": "openstac",
      "port": "50000",
      "user": "dash"
    }
  },
  "db2": {
    "das": {
      "password": "passw0rd"
    },
    "fenced": {
      "password": "passw0rd"
    },
    "instance": {
      "password": "passw0rd"
    }
  }
}

Chapter 2. Installing  163
Requirements for upgrading a VMware Region

There are some requirements for upgrading a VMware Region from SmartCloud Orchestrator V2.3.x to IBM Cloud Orchestrator V2.4.

About this task

SmartCloud Orchestrator V2.3.x supports an EXSI under the vCenter (DataCenter) directly, but IBM Cloud Orchestrator V2.4 only support clusters under the vCenter (DataCenter) officially. This means that IBM Cloud Orchestrator V2.4 only discovers a cluster as its hypervisor. Then all EXSIs must be under some clusters in vCenter if you want to upgrade a VMware region from SmartCloud Orchestrator V2.3.x to IBM Cloud Orchestrator V2.4.

Procedure

1. Create clusters in vCenter.
2. Move all EXSIs which are not under clusters to a certain cluster.
3. Wait a few minutes for the synchronization from vCenter to SmartCloud Orchestrator V2.3.x.
4. Upgrade the VMware Region.
5. Manually migrate these virtual machines which are under the EXSIs.
Migrating data in a VMware region

In SmartCloud Orchestrator V2.3, you can use an sce driver to manage VMware. In IBM Cloud Orchestrator V2.4 an OpenStack native driver replaces it. There is a ds upgrade tool to upgrade the VMware region from SmartCloud Orchestrator V2.3 to IBM Cloud Orchestrator V2.4. However, in some cases, you must perform some manual work to migrate the data.

Procedure
1. Migrate the Availability Zone:
   a. Discover the Availability Zone in your Region node:
      
      ```
      [root@scotest-d20-region1 ~]# nova availability-zone-list
      +-----------------------------+----------------------------------------+
      | Name | Status |
      +-----------------------------+----------------------------------------+
      | internal | available |
      | |- scotest-d20-region1 | |
      | | |- nova-cert | enabled :-) 2014-07-19T02:21:49.041585 |
      | | |- nova-conductor | enabled :-) 2014-07-19T02:21:45.707118 |
      | | |- nova-consoleauth | enabled :-) 2014-07-19T02:21:40.908027 |
      | | |- nova-network | enabled :-) 2014-07-19T02:21:40.510854 |
      | | |- nova-scheduler | enabled :-) 2014-07-19T02:21:46.600970 |
      | | |- nova-smartcloud | enabled :-) 2014-07-19T02:21:44.159949 |
      | Cluster1@172.19.4.11-443 | available |
      | |- Cluster1@172.19.4.11-443 | |
      | | |- nova-compute | enabled :-) 2014-07-19T02:21:44.171048 |
      +-----------------------------+----------------------------------------+
      ```

      There are two Availability Zones: internal and Cluster1@172.19.4.11-443. The later includes a service nova-compute, and then it is the Availability Zone that manages your virtual machines (these virtual machines that are under the cluster cluster1 in vCenter).

      **Note:** There might be more than one Availability Zones that manage some virtual machines in your Region. Each of them are in format cluster_name@vCenter_ip. Each Availability Zone “cluster_name@vCenter_ip” manage these virtual machines that are under the cluster_name in vCenter. You can check it in vCenter.

2. Create new configuration files for the Availability Zones:
   You can create new configuration files by copying your /etc/nova/nova.config.
   Take the result of Step 1 as an example:
   ```
   cp /etc/nova/nova.config /etc/nova/nova_cluster1_172.19.4.11.conf
   ```
   Then edit the /etc/nova/nova.config /etc/nova/
   nova_cluster1_172.19.4.11.conf. You can run the following commands:
   ```
   openstack-config --set /etc/nova/nova_cluster1_172.19.4.11.conf DEFAULT default_availability_zone Cluster1@172.19.4.11-443
   openstack-config --set /etc/nova/nova_cluster1_172.19.4.11.conf DEFAULT default_schedule_zone Cluster1@172.19.4.11-443
   openstack-config --set /etc/nova/nova_cluster1_172.19.4.11.conf DEFAULT storage_availability_zone Cluster1@172.19.4.11-443
   openstack-config --set /etc/nova/nova_cluster1_172.19.4.11.conf DEFAULT host Cluster1@172.19.4.11-443
   openstack-config --set /etc/nova/nova_cluster1_172.19.4.11.conf vmware cluster_name Cluster1
   openstack-config --set /etc/nova/nova_cluster1_172.19.4.11.conf vmware host_ip 172.19.4.11
   ```

3. Create new service files for the Availability Zones:
   You can create new configuration files by copying your /etc/init.d/
   openstack-nova-compute. For example:
   ```
   cp /etc/init.d/openstack-nova-compute /etc/init.d/openstack-nova-compute-Cluster1-172.19.4.11
   ```
   Then apply the configuration file created in Step 2, by changing:
4. Start your created service files:

   service openstack-nova-compute-Cluster1-172.19.4.11 start

5. Clean the hypervisor:

   a. Find the hypervisors:

      ```
      [root@scotest-d20-region1 init.d]# nova hypervisor-list
      +----+--------------------------+
      | ID | Hypervisor hostname      |
      +----+--------------------------+
      | 1  | Cluster1@172.19.4.11-443 |
      | 21 | domain-c7(Cluster1)      |
      | 22 | domain-c7(Cluster1)      |
      +----+--------------------------+
      ```

      The Cluster1@172.19.4.11-443 is the original hypervisor name in SmartCloud Orchestrator V2.3. The domain-c7(Cluster1) with id 21 is created by openstack-nova-compute, it belongs to the default Availability Zone nova. The domain-c7(Cluster1) with id 22 is created by your created /etc/init.d/openstack-nova-compute-Cluster1-172.19.4.11, it belongs to the created Availability Zone Cluster1@172.19.4.11-443. The hypervisor name with id 1 and 2 must be deleted and the hypervisor name with id 22 must change its id to 1 (the original id).

      Go to the DB2 server:

      ```
      db2 "delete db_username.compute_nodes where id=1"
      db2 "update db_username.compute_nodes set id=1 where id=22"
      db2 "delete db_username.compute_nodes where hypervisor_hostname='domain-c7(Cluster1)' and id=1"
      ```

   b. Migrate the instance:

      For a certain instance, you must know which cluster it belongs to. You can check it by running `nova show`. If its properties such as Availability Zone, hypervisor_name are in the format of cluster_name@vCenter it belongs to the cluster of cluster_name. If it originally belongs to an EXSI which is not in a cluster, you must create a cluster and move the EXSI into it (in SmartCloud Orchestrator V2.3 phrase). In IBM Cloud Orchestrator V2.4, you might need to migrate the date manually. For example, an instance belongs to cluster Cluster1 in the vCenter 172.19.4.11, you must change its properties in the DB2 server. Go to the DB2 server:

      ```
      db2 "update db_username.instances set node='domain-c7(Cluster1)', vm_state='active', host='Cluster1@172.19.4.11-443' where uuid='instance_uuid'"
      ```

      Note: You can find the value of node by running `nova hypervisor-list`. The item that contains the cluster name is the one that you want.

   c. Migrate the template:

      Go to DB2 server and create the file:

      `/home/db2inst1/vmware_template_data_migration.sql`

      Connect to OpenStack:

      ```
      export to /home/db2inst1/template_data_migration-20140718134138.backup of DEL lobfile template_data_migration-20140718134138.lob modified by lobss
      merge into gle20918."IMAGE_PROPERTIES" prop \
      using (select id as image_id, 'template_name' as name_key, name as name_value, deleted \ 
      from (select distinct id, name, deleted from gle20918."IMAGES" \ 
      where status in ('active', 'deleted')) image \ 
      on prop.image_id = image.image_id and prop.name = image.name_key \ 
      when not matched \ 
      then insert (image_id, name, "value", created_at, deleted) values \
      (image.image_id, image.name_key, image.name_value, current timestamp, image.deleted)
      ```

      Then run the following command:
su - db2inst1 -c 'db2 -w- -v -v /home/db2inst1/vmware_template_data_migration.sql'

d. Migrate the network:
  Go to the DB2 Server and create the file:
  /home/db2inst1/nova_network_migration.sql:
  Connect to OpenStack.
  Export to '/home/db2inst1/nova_network_migration-20140718134138.backup' of DEL LOBFIL
  E network_lobs select * from noa20918.NETWORKS.
  merge into noa20918.NETWORKS nova 
  using (select "networkName" as netname, "osNetworkId" as netid 
  from sce20918.NETWORK_EXTENSIONS) sce 
  on nova.uuid = sce.netid 
  when matched then 
    update set nova."label"=sce.netname

  Then run the command:
  su - db2inst1 -c 'db2 -w- -v -v /home/db2inst1/nova_network_migration.sql'

Upgrade troubleshooting
Continue the upgrade if the upgrade job failed

About this task

If upgrade job failed due to wrong configurations in the upgrade configuration file, you can roll back the IBM Cloud Orchestrator servers, correct the configuration, and rerun the ico-upgrade-tool to fix your problems. The IBM Cloud Orchestrator servers can be roll backed according to "Configuring the upgrade configuration file" on page 159.

1. If Central Server upgrade job failed, rollback all Central Servers. You do not need to roll back the Region Servers because the Region Server upgrade job does not execute until the Central Server upgrade jobs are finished successfully.

2. If Central Server upgrade job is finished successfully, but the Region Server upgrade jobs failed due to wrong configurations in the upgrade configuration file, rollback the Region Servers that failed to run the upgrade jobs.

Procedure

1. Roll back the IBM Cloud Orchestrator Servers according to what indicated in About this task.
2. Go to the location of the upgrade configuration file and edit the configuration file with the correct configurations.
3. Run the command ico-upgrade-tool upgrade -C CENTRAL_SRV1_IP -p CENTRAL_SRV1_ROOT_PASS to rerun the failed upgrade jobs. ico-upgrade-tool bypasses the finished upgrade jobs.
4. Check the job status to make sure that the upgrade job finished successfully.

Vmware region failed to start nova-network after upgrade

About this task:

If nova-network failed to start after upgrade with following error, please following this troubleshooting to make nova-network works:
novas.openstack.common.threadgroup \nStderr: "iptables-restore v1.4.7: host/network 'None' not found\nError occurred at line: 27\nTry `iptables-restore -h' or 'iptables-restore --help' for more information."

Procedure:
1. Backup /etc/sysconfig/iptables.
2. Clean iptables with the command:
   iptables -F & & iptables -t nat -F
   service iptables save
3. Restart the nova network service for the special cluster, for example, if your
   cluster name sco-cluster-01, restart nova network service with following
   command:
   service openstack-nova-network-sco-cluster-01 restart
4. Check the backup file if there is additional rules need to be added manually.

Upgrading from SmartCloud Cost Management 2.1.0.3
You can upgrade from SmartCloud Cost Management 2.1.0.3 to SmartCloud Cost
Management 2.1.0.4 by using the sccm_install.sh script. The upgrade is provided
as a console mode upgrade only.

Before you begin
Ensure that the system that you are upgrading is the same system where
SmartCloud Cost Management 2.1.0.3 has been installed. IBM Cloud Orchestrator
V2.3 or V2.3.0.1 must be successfully upgraded to IBM Cloud Orchestrator V2.4.
The properties defined in the sccm_install.properties file during the installation
of SmartCloud Cost Management 2.1.0.3 must also be used for the 2.1.0.4 upgrade.
These properties include:

INSTALL_DIR
The installation directory of SmartCloud Cost Management. This directory
must be the same as that used in 2.1.0.3 as the sccm_install.sh script will
use this to detect if there is a previous install and therefore if it needs to
run in upgrade mode.

ADMIN_USER
The SmartCloud Cost Management administration user.

ADMIN_PASSWORD
The SmartCloud Cost Management administration password.

HTTPS_PORT
The https port which should be used by SmartCloud Cost Management.

Note: The HTTP_PORT property is no longer present in 2.1.0.4 as SmartCloud
Cost Management can now only be configured to use https.

Note: Once IBM Cloud Orchestrator V2.3 has been upgraded to IBM Cloud
Orchestrator V2.4, the SmartCloud Cost Management 2.1.0.4 upgrade should
happen immediately afterwards to prevent the loss of any metering records.

Procedure
1. Log in to the system using the login credentials required for installing on a
   Linux platform.
2. Enter one of the following commands:
   • ./sccm_install.sh
3. Follow the directions presented on the screen to complete the installation.

4. Launch the browser: https://host:port/Blaze/Console. For example, https://servername:9443/Blaze/Console

5. Once installation is successfully completed, run the automated post-installation configuration to ensure that SmartCloud Cost Management is configured to work with IBM Cloud Orchestrator 2.4.

Note: Metering and Billing will not work correctly for an upgraded IBM Cloud Orchestrator 2.4 environment unless configured correctly to do so.

Note: If you have modified job files that refer to the 2.1.0.3 default Database datasource, after running the automated post-installation configuration process, you must change the name of the datasource in these job files to the new datasource name used in 2.1.0.4. The name of the default Database datasource in 2.1.0.3 is sco_db2 and in 2.1.0.4 is ico_db2.

### Post configuration after Public Cloud Gateway is upgraded

Perform the following task after Public Cloud Gateway is upgraded.

During an upgrade from SmartCloud Orchestrator 2.3 to IBM Cloud Orchestrator 2.4, the Public Cloud Gateway configuration file flavors.json (located by default in /opt/ibm/pcg/etc) is updated. Any preexisting version of the file is backed up and stored in the flavors.json.bak file in the same directory. The backup file can be used to retain any changes you made to the original file.

If you change the flavors.json file, you must restart the Public Cloud Gateway by running the following command:

```bash
service pcg restart
```

### Advanced post configuration after a VMware region is upgraded

If you are using datastore clusters or resource pools on your VMware region managed to vCenter, perform the following procedure to configure your VMware region after upgrading from SmartCloud Orchestrator 2.3.

#### About this task

In the following procedure, Cluster1 is a cluster on vCenter, and ResourcePool1 is a resource pool under this cluster.

**Note:**

- If you have multiple clusters and resource pools, you must perform the following procedure for each of them.
- If you are using both datastore cluster and resource pool, perform all the steps of the following procedure.
- If you are using only datastore cluster, perform steps 2, 4, and 7 of the following procedure.
- If you are using only resource pool, perform the following procedure by skipping step 2.
Procedure

1. Run the following command to get a list of existing hypervisors and record the related hypervisor IDs, because the original hypervisor is deleted by the nova compute after the manual configuration and after restarting the nova compute service.

   ```
   nova hypervisor-list
   ```

   +-----+-----------------------------------------------------+
   | ID  | Hypervisor hostname                                 |
   +-----+-----------------------------------------------------+
   | 101 | domain-c27(Cluster1)                                |
   | 102 | resgroup-87(ResourcePool1)                           |
   +-----+

2. Configure the datastore cluster:
   a. Edit each nova configuration file of VMware cluster and resource pool, for example the `/etc/nova/nova-192.0.2.56-443_Cluster1.conf` and `/etc/nova/nova-192.0.2.56-443_ResourcePool1.conf` files.
   b. Add

      ```
      datastore_cluster_name = DatastoreCluster1
      ```

      under datastore_regex and remove the datastore_regex attribute.

      **Note:** You can configure only one cluster for datastore_cluster_name. Regular expression for this option is not supported.

3. Configure the resource pool by editing the nova configuration file of the resource pool. Note that if the nova compute service is connecting to a real cluster as in this procedure, you do not need to update the nova configuration file of the cluster.

   The resource pool can be under an ESXi host or under a real cluster. If the resource pool is under a real cluster, perform the following steps:
   a. Change the following attribute:

      ```
      cluster_name = <resource pool name>
      ```

      to

      ```
      cluster_name = <the cluster name under which this resource pool is>
      ```

   b. Under the cluster_name attribute, add the following attribute:

      ```
      resource_pool = <the cluster name under which this resource pool is>:<resource pool name>
      ```

      If the resource pool is under an ESXi host, perform the following steps:
   a. Under the cluster_name attribute, add the following attribute:

      ```
      resource_pool = <the ESXi host name under which this resource pool is>:<resource pool name>
      ```

   b. Delete the cluster_name option.

   For example, if the resource pool is under a real cluster, you must specify the following attributes in the `/etc/nova/nova-192.0.2.56-443_ResourcePool1.conf` file:

   ```
   cluster_name = Cluster1
   resource_pool = Cluster1:ResourcePool1
   ```

4. Restart the nova services related to the cluster or resource pool by running the following commands, for example:

   ```
   /etc/init.d/openstack-nova-compute-Cluster1 restart
   /etc/init.d/openstack-nova-network-Cluster1 restart
   /etc/init.d/openstack-nova-compute-ResourcePool1 restart
   /etc/init.d/openstack-nova-network-ResourcePool1 restart
   ```
Check the related service status until the status is :-) by running the following command:

```
nova availability-zone-list
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourcePool1@vcell10-443</td>
<td>available</td>
</tr>
<tr>
<td>Cluster1@vcell10-443</td>
<td>available</td>
</tr>
<tr>
<td>nova-network</td>
<td>enabled :-) 2014-08-29T02:39:00.199581</td>
</tr>
<tr>
<td></td>
<td>nova-network</td>
</tr>
<tr>
<td>test-srv05</td>
<td>enabled :-) 2014-08-29T10:36:23.328309</td>
</tr>
<tr>
<td>nova-compute</td>
<td>enabled :-) 2014-08-29T10:36:30.183126</td>
</tr>
<tr>
<td>nova-cert</td>
<td>enabled :-) 2014-08-29T10:36:22.339324</td>
</tr>
<tr>
<td>nova-conductor</td>
<td>enabled :-) 2014-08-29T10:36:25.947820</td>
</tr>
<tr>
<td>nova-consoleauth</td>
<td>enabled :-) 2014-08-29T10:36:25.598537</td>
</tr>
<tr>
<td>nova-network</td>
<td>enabled :-) 2014-08-29T10:36:25.947820</td>
</tr>
<tr>
<td>nova-scheduler</td>
<td>enabled :-) 2014-08-29T10:36:25.947820</td>
</tr>
<tr>
<td>nova-smartcloud</td>
<td>enabled :-) 2014-08-29T10:36:25.947820</td>
</tr>
<tr>
<td>Cluster1@vcell10-443</td>
<td>available</td>
</tr>
<tr>
<td>nova</td>
<td>available</td>
</tr>
<tr>
<td>test-srv05</td>
<td>enabled :-) 2014-08-29T10:36:29.350317</td>
</tr>
<tr>
<td>nova-compute</td>
<td>enabled :-) 2014-08-29T10:36:29.350317</td>
</tr>
</tbody>
</table>

You can see the new hypervisor by using the `nova hypervisor-list` command. In the output the hypervisor name is the same as the resource pool specified in the nova configuration file. For example:

```
| 178 | Cluster1:ResourcePool1 |
```

5. To ensure that the hypervisor ID is not changed after the manual configuration, you must run the `change_compute_id.sh` script on the VMware region node. For example, if your original hypervisor ID is 102:

```
| 102 | resgroup-87(ResourcePool1) |
```

after the manual configuration, 102 was marked as deleted in database. You can only see the new hypervisor 178 by using the `nova hypervisor-list` command:

```
| 178 | Cluster1:ResourcePool1 |
```

**Note:** The original hypervisor 102 is not shown by using the `nova hypervisor-list` command after the manual configuration. You must record this ID before starting this procedure.

Log on to your VMware region node, and run following command:

```
cd /opt/ibm/orchestrator/hypervisor_tool
./change_compute_id.sh --org_id 178 --new_id 102
```

**Note:** If resgroup- is a prefix of the hypervisor name, it must be a resource pool in vCenter. You must follow this post configuration step to configure it, otherwise you can not boot the virtual machine successfully.

6. Restart the nova services related to the resource pool and the nova scheduler service by running the following commands:

```
/etc/init.d/openstack-nova-compute-ResourcePool1 restart
/etc/init.d/openstack-nova-network-ResourcePool1 restart
/etc/init.d/openstack-nova-scheduler restart
```
Check the related services status until the status is :)

7. Try to run OpenStack CLI to boot the virtual machine to ensure that OpenStack works.
   Log on the VMware region and try the following commands to boot virtual machines:
   ```bash
   source /root/openrc
   nova boot --flavor 1 --image <your image id> --availability-zone ResourcePool1@vcell10-443 testenv1
   ```
   Check that the virtual machines status to ensure it is active.

---

**Ports used by IBM Cloud Orchestrator**

For information about port management and security, see the [IBM Cloud Orchestrator Security Hardening Guide](#)
Chapter 3. Administering

After you have installed IBM Cloud Orchestrator, you can start your environment, configure optional settings, and define users, projects, and domains as described in the following sections.

Starting or stopping IBM Cloud Orchestrator

You can start or stop IBM Cloud Orchestrator.

Before you begin

This process does not apply to Distributed Active-Active (High Availability) installations. For details of how to start or stop IBM Cloud Orchestrator in high-availability installations, see “Controlling the management stack” on page 195.

Before you run the SCOrchestrator.py script, review Managing services with SCOrchestrator.py and ensure that your environment satisfies the conditions for running the script.

About this task

You can run the SCOrchestrator.py script to start, stop, and view the status of the IBM Cloud Orchestrator components.

For information about running the SCOrchestrator.py script, see “Running the SCOrchestrator.py script” on page 177.

If you are using System Automation Application Manager, IBM Cloud Orchestrator must only be started and stopped via System Automation Application Manager and must not be managed by the SCOrchestrator.py script. For more information, see “Controlling the management stack” on page 195.

To completely stop the IBM Cloud Orchestrator system:
1. Run the SCOrchestrator.py script to stop all services. For information about running the SCOrchestrator.py script, see “Running the SCOrchestrator.py script” on page 177.
2. Stop your virtual machines in no specific order.

To start the IBM Cloud Orchestrator system when nodes are down:
1. Start all the virtual machines in no particular order.
2. Run the SCOrchestrator.py script to stop all services. Check that all services have stopped.
3. Run the SCOrchestrator.py script to start all services.
4. Restart NoSQL by following the procedure described in “Restarting NoSQL” on page 187.
Accessing IBM Cloud Orchestrator user interfaces

IBM Cloud Orchestrator provides several user interfaces to access the various components.

To correctly display the IBM Cloud Orchestrator user interfaces, use one of the following browsers:
- Internet Explorer versions 10, and 11
- Firefox Extended Support Release 31
- Google Chrome version 37

Access to the various IBM Cloud Orchestrator user interfaces depends on the role assigned, as shown in the following table:

<table>
<thead>
<tr>
<th>User interface</th>
<th>URL</th>
<th>Access granted to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration user interface</td>
<td>https://central-server-ihs_fqdn</td>
<td>admin role only</td>
</tr>
<tr>
<td>Business Process Manager user</td>
<td>https://central-server-ihs_fqdn:8443/ProcessCenter/login.jsp</td>
<td>admin role only</td>
</tr>
<tr>
<td>self-service user interface</td>
<td>https://central-server-ihs_fqdn:8443</td>
<td>admin role, domain_admin role, catalogeditor role, member role</td>
</tr>
</tbody>
</table>

Each IBM Cloud Orchestrator user interface URL includes the following specification:
https://central-server-ihs_fqdn

where central-server-ihs_fqdn is the fully qualified DNS domain name of the Central Server where the IBM HTTP Server is installed. In a multiple-region deployment, the IBM HTTP Server is installed on Central Server 2.

**Note:**
- The fully qualified DNS domain name is in the form x.y.z, as shown in the following example:
  host.example.com

  **Remember:** Replace the example value with the name of your DNS domain.
- Do not use the IP address to access the IBM Cloud Orchestrator user interfaces.
- When logging to the Administration user interface, if you save the password details with the browser, these details might be loaded unexpectedly in the Update User dialog for that user. To avoid this behaviour, you must clear the password details from the browser.
- If you are using the Administration user interface in non-English language, you might see English strings because of OpenStack limitations.

You can extend the Self-service user interface URL to include the IBM Cloud Orchestrator domain name, as shown in the following example:
https://central-server-ihs_fqdn:8443/login?domainName=myDomain
In this example, the **Domain** field on the login screen is prepopulated with the value `myDomain`. If you do not specify a domain, the user is authenticated to the **Default** domain.

By default, the user is also authenticated to the scope of the primary project that you specified when you created the user. After you log in, you can change the project scope by selecting a new project from the project list in the top banner of the user interface. For more information about users, projects, and domains, see "Managing security" on page 203.

**Note:** In IBM Cloud Orchestrator, the following limitations apply:

- A user name cannot contain a colon (:) character.
- A password cannot contain an at sign (@) character.
- Users cannot log in if the primary project to which they are assigned is disabled.
- You cannot log in to the same IBM Cloud Orchestrator user interface with more than one browser session on the same machine. If you must log in to the same IBM Cloud Orchestrator user interface with two browser sessions on the same machine, use a different browser for each session. For example, use an Internet Explorer browser and a Firefox browser.
- The Administration user interface login credentials are case-sensitive.

To view the IBM Cloud Orchestrator user interface in another language, set the language option in your browser. Move the preferred language to the top of the list, clear the browser cache, and refresh your browser view. For some browser and operating system combinations, you might need to change the regional settings of your operating system to the locale and language of your choice.

You must set the locale in Business Process Manager separately. Log into the Business Process Manager user interface, and click **Preferences**. Select the locale from the **Locale preferences** list, and click **Save changes**. You might need to log in again for the changes to take effect.

---

**Managing the services**

Understand how to manage the IBM Cloud Orchestrator services.

**Managing services with SC0rchestrator.py**

The **SC0rchestrator.py** Python script can start, stop, and provide information about the status of IBM Cloud Orchestrator services.

The **SC0rchestrator.py** script is installed on Central Server 1, in the `/opt/ibm/orchestrator/scorchestrator` directory. If you install IBM Cloud Orchestrator as highly available, the **SC0rchestrator.py** script is not installed. Instead, you use System Automation Application Manager to manage the IBM Cloud Orchestrator services. For more information, see "Controlling the management stack" on page 195.

IBM Cloud Orchestrator contains of a number of services and modules, which have to be online or running before the product can be used. The modules and services are spread across several virtual appliances. Because some of these modules and services require being started and stopped in sequence, IBM Cloud Orchestrator is delivered with the **SC0rchestrator.py** script to start, stop, and monitor the status of a single component or to start and stop all the IBM Cloud Orchestrator services with one command. If you use the **SC0rchestrator.py** script
to start or stop all the IBM Cloud Orchestrator services, the services are started or stopped in the correct sequence and all the dependencies are resolved.

The SCOrchestrator.py script uses XML files to obtain the information about the environment and the components:
- SCOEnvironment.xml
- SCOComponents.xml
- SCOEnvironment_fulltopo.xml

The XML files define the names and the start or stop priority of the IBM Cloud Orchestrator services.

The SCOEnvironment.xml file is automatically generated by the installation procedure when the central IBM Cloud Orchestrator servers are installed. Afterwards, the installation procedure automatically updates the file if a Region Server is installed. You must manually modify the file only if a Region Server is removed.

The following sequence occurs when you run SCOrchestrator.py:
1. The script invokes SCOEnvironment_update.py to refresh the region topology and generate or update the SCOEnvironment_fulltopo.xml file.
2. The script reads the SCOEnvironment_fulltopo.xml file, or the SCOEnvironment.xml file if the SCOEnvironment_fulltopo.xml file does not exist, and the SCOComponents.xml file.
3. The script obtains the parameters and options and analyzes them for the actions to take.
4. For start, stop, and status sequences, scripts are copied to the specific systems and executed on the remote systems.
5. The scripts are cleaned up from the /tmp directory of the systems.

The script files which are executed on the systems are in the same directory as the SCOrchestrator.py script.

You can start or stop certain components if you know the exact name of the component or host name. You can start or stop all modules of a specific virtual machine by using the host name of the machine. As a result, all components of that machine are started or stopped.

Note: Because some IBM Cloud Orchestrator services must be started in a given sequence to work correctly, do not start or stop any single services but only start or stop the whole IBM Cloud Orchestrator stack. Only experienced administrators who are aware of the dependencies between the IBM Cloud Orchestrator services can use the SCOrchestrator.py script to start or stop single services.

Assumptions for running the SCOrchestrator.py script
- The script is executed as root on the Central Server 1.
- Components of additional compute nodes are hardcoded.
- No Windows operating system support on management systems.
Running the SCOrchestrator.py script

You can run the SCOrchestrator.py script to start, stop and view the status of the IBM Cloud Orchestrator components.

Procedure
1. As root, navigate to /opt/ibm/orchestrator/scorchestrator on the Central Server 1.
2. Run the SCOrchestrator.py script with the following options:
   • To start the whole product, run ./SCOrchestrator.py --start.
   • To stop the whole product, run ./SCOrchestrator.py --stop.
   • To view the status of components, run ./SCOrchestrator.py --status.
   • To view help for this script, run ./SCOrchestrator.py --help. The following help is displayed:
     Usage: SCOrchestrator.py [options]
     Options:
     -h, --help shows this help message and exit
     -s, --start starts IBM Cloud Orchestrator with default parameters in default sequence
     --halt, --shutdown, --stop stops IBM Cloud Orchestrator in default sequence
     --status shows status of components of IBM Cloud Orchestrator
     -c SCOComponents.xml, --componentfile=SCOComponents.xml defines the input properties filename. Default name is SCOComponents.xml
     -e SCOEnvironment.xml, --environmentfile=SCOEnvironment.xml defines the environment filename. Default name is SCOEnvironment.xml
     -n SYSTEMSLIST, --hostips=SYSTEMSLIST list of host IPs to start/stop format hostip1,hostip2,hostip3,...
     -p COMPONENTSLIST, --components=COMPONENTSLIST list of components to start/stop format component1,component2,component3,...
     --version shows SCOrchestrator.py version and exits

   To view a list of components that can be specified with the -p option, open the SCOEnvironment_fulltopo.xml file that lists all available component names for each host of the IBM Cloud Orchestrator environment.

Running the SCOrchestrator.py script with non-root user

The following commands are all run as root on the server indicated.

Procedure
1. Create a new user ssh for all the Central Servers and Region Servers:
   a. On each of the Central Servers and Region Servers:
      Create a new user <yourmechid> and set the password:
      useradd -m <yourmechid>
pwd <yourmechid>  #enter at the prompt <yourmechpwd>

      Create an .ssh directory and set file permissions:
      su - <yourmechid> -c "mkdir .ssh; chmod 700 .ssh"

   b. On Central Server 1:
      Generate the ssh keys for <yourmechid> and copy it to all IBM Cloud Orchestrator servers:
      su - <yourmechid> -c "ssh-keygen -q -t rsa -N '' -f "/.ssh/id_rsa""
Here $i$ stands for the IP address of each IBM Cloud Orchestrator server including Central Server 1:

```
[root@cs-1] su <yourmechid>
[<yourmechid@cs-1] scp ~/.ssh/id_rsa.pub $i:~/.ssh/authorized_keys
```

**Note:** It is important in the command above that you accept the server key and the password required is of <yourmechid>.

c. Verify that <yourmechid> on Central Server 1 can ssh to all the IBM Cloud Orchestrator servers including central Server 1 without interruption:

```
su - <yourmechid> -c "ssh <yourmechid>@$SCO_server_ip"
su - <yourmechid> -c "ssh <yourmechid>@$SCO_server_ip"
```

2. Copy the /opt/ibm/orchestrator and change the directory permission:

On Central Server 1, copy the directory /opt/ibm/orchestrator to /home/<yourmechid>:

```
cp -rf /opt/ibm/orchestrator/scorchestrator /home/<yourmechid>
```

Change the owner of /home/<yourmechid>:

```
chown -R <yourmechid>:<yourmechidgroup> /home/<yourmechid>/
```

3. On each of the IBM Cloud Orchestrator servers, add the user <yourmechid> in the sudo list:

a. Create a sudoer file named <yourmechid> and place it in /etc/sudoers.d.

   The content of the file <yourmechid> is like what follows:

   **Note:** Replace <yourmechid> with your new user name.

   ```
   # sudoers additional file for /etc/sudoers.d/
   # IMPORTANT: This file must have no ~ or . in its name and file permissions # must be set to 440!!!
   # this file is for the SAAM mech-ID to call the SCO control scripts
   Defaults:<yourmechid> !requiretty
   # scripts found in control script directory
   # adapt the directory names to the mech id!
   Cmnd_Alias SACTRL = /tmp/*.sh
   # allow for <yourmechid> ALL = (root) NOPASSWD: SACTRL
   
   b. Change the sudoer file permission:

   ```chmod 440 <yourmechid>```

4. On Central Server 1, run these commands from the command prompt:

   ```
   sed -i "s/\./sudo \./g" /home/<yourmechid>/scorchestrator/SCOrchestrator.py
   sed -i "s/\./sudo \./g" /home/<yourmechid>/scorchestrator/SCOEnvironment_update.py
   sed -i -e 's/"SSH_USER=\*SSH_USER = "<yourmechid>"/g' 
   /home/<yourmechid>/scorchestrator/SCOrchestrator.py
   sed -i -e 's/"SSH_USER=\*SSH_USER = "<yourmechid>"/g' 
   /home/<yourmechid>/scorchestrator/SCOEnvironment_update.py
   ```

5. Run SCOrchestrator.py:

   Move to the path /home/<yourmechid>/scorchestrator and run SCOrchestrator.py with user <yourmechid>:

   ```
   [root@cs-1] cd /home/<yourmechid>/scorchestrator
   [root@CS-1 scorchestrator]# su <yourmechid>
   [yourmechid@CS-1 scorchestrator]$ ./SCOrchestrator.py
   ```

   For the usage of SCOrchestrator.py, refer to “Running the SCOrchestrator.py script” on page 177.
Managing IBM Cloud Orchestrator services manually

In a standard multiple region deployment environment, you can check and start the services of IBM Cloud Orchestrator.

In general, to manage services you use $SCOrchestrator.py; however, if you are an advanced user you can manage services manually.

The following table illustrates how you can check and start the services of IBM Cloud Orchestrator.

Note: If you are using an all-in-one topology, all the services run on the same computer.

<table>
<thead>
<tr>
<th>Server name</th>
<th>Components deployed</th>
<th>Command to check the service</th>
<th>Command to start the service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Server 1</td>
<td>DB2</td>
<td>ps -ef</td>
<td>grep db2, check port 5000</td>
</tr>
<tr>
<td></td>
<td>ceilometer-collector</td>
<td>service openstack-ceilometer-collector status</td>
<td>service openstack-ceilometer-collector start</td>
</tr>
<tr>
<td></td>
<td>ceilometer-agent-central</td>
<td>service openstack-ceilometer-agent-central status</td>
<td>service openstack-ceilometer-agent-central start</td>
</tr>
<tr>
<td>Central Server 2</td>
<td>Business Process Manager</td>
<td>ps -ef</td>
<td>grep bpm</td>
</tr>
<tr>
<td></td>
<td>Self-service user interface</td>
<td>ps -ef</td>
<td>grep scui</td>
</tr>
<tr>
<td></td>
<td>Public Cloud Gateway</td>
<td>ps -ef</td>
<td>grep pcg</td>
</tr>
<tr>
<td></td>
<td>Keystone</td>
<td>service openstack-keystone status</td>
<td>service openstack-keystone start</td>
</tr>
<tr>
<td></td>
<td>ceilometer-api</td>
<td>service openstack-ceilometer-api status</td>
<td>service openstack-ceilometer-api start</td>
</tr>
<tr>
<td></td>
<td>Administration user interface</td>
<td>service httpd status</td>
<td>service httpd start</td>
</tr>
<tr>
<td></td>
<td>IBM HTTP server</td>
<td>service ihs status</td>
<td>service ihs start</td>
</tr>
<tr>
<td>Central Server 3</td>
<td>Workload Deployer</td>
<td>service iwd status</td>
<td>service iwd start</td>
</tr>
</tbody>
</table>
### Table 15. Managing IBM Cloud Orchestrator services manually (continued)

<table>
<thead>
<tr>
<th>Server name</th>
<th>Components deployed</th>
<th>Command to check the service</th>
<th>Command to start the service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region Server</td>
<td>nova-api</td>
<td>service openstack-nova-api status</td>
<td>service openstack-nova-api start</td>
</tr>
<tr>
<td></td>
<td>nova-scheduler</td>
<td>service openstack-nova-scheduler status</td>
<td>service openstack-nova-scheduler start</td>
</tr>
<tr>
<td></td>
<td>nova-conductor</td>
<td>service openstack-nova-conductor status</td>
<td>service openstack-nova-conductor start</td>
</tr>
<tr>
<td></td>
<td>glance-api</td>
<td>service openstack-glance-api status</td>
<td>service openstack-glance-api start</td>
</tr>
<tr>
<td></td>
<td>glance-registry</td>
<td>service openstack-glance-registry status</td>
<td>service openstack-glance-registry start</td>
</tr>
<tr>
<td></td>
<td>cinder-api</td>
<td>service openstack-cinder-api status</td>
<td>service openstack-cinder-api start</td>
</tr>
<tr>
<td></td>
<td>cinder-volume</td>
<td>service openstack-cinder-volume status</td>
<td>service openstack-cinder-volume start</td>
</tr>
<tr>
<td></td>
<td>cinder-scheduler</td>
<td>service openstack-cinder-scheduler status</td>
<td>service openstack-cinder-scheduler start</td>
</tr>
<tr>
<td></td>
<td>heat-api</td>
<td>service openstack-heat-api status</td>
<td>service openstack-heat-api start</td>
</tr>
<tr>
<td></td>
<td>heat-api-cfn</td>
<td>service openstack-heat-api-cfn status</td>
<td>service openstack-heat-api-cfn start</td>
</tr>
<tr>
<td></td>
<td>heat-api-cloudwatch</td>
<td>service openstack-heat-api-cloudwatch status</td>
<td>service openstack-heat-api-cloudwatch start</td>
</tr>
<tr>
<td></td>
<td>heat-engine</td>
<td>service openstack-heat-engine status</td>
<td>service openstack-heat-engine start</td>
</tr>
<tr>
<td></td>
<td>ceilometer-notification</td>
<td>service openstack-ceilometer-notification status</td>
<td>service openstack-ceilometer-notification start</td>
</tr>
<tr>
<td></td>
<td>ceilometer-collector</td>
<td>service openstack-ceilometer-collector status</td>
<td>service openstack-ceilometer-collector start</td>
</tr>
<tr>
<td></td>
<td>ceilometer-agent-central</td>
<td>service openstack-ceilometer-agent-central status</td>
<td>service openstack-ceilometer-agent-central start</td>
</tr>
<tr>
<td></td>
<td>neutron-server</td>
<td>service neutron-server status</td>
<td>service neutron-server start</td>
</tr>
</tbody>
</table>
### Table 15. Managing IBM Cloud Orchestrator services manually (continued)

<table>
<thead>
<tr>
<th>Server name</th>
<th>Components deployed</th>
<th>Command to check the service</th>
<th>Command to start the service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutron Network Node</td>
<td>neutron-linuxbridge-agent</td>
<td>service neutron-linuxbridge-agent status</td>
<td>service neutron-linuxbridge-agent start</td>
</tr>
<tr>
<td></td>
<td>neutron-l3-agent</td>
<td>service neutron-l3-agent status</td>
<td>service neutron-l3-agent start</td>
</tr>
<tr>
<td></td>
<td>neutron-metadata-agent</td>
<td>service neutron-metadata-agent status</td>
<td>service neutron-metadata-agent start</td>
</tr>
<tr>
<td></td>
<td>neutron-dhcp-agent</td>
<td>service neutron-dhcp-agent status</td>
<td>service neutron-dhcp-agent start</td>
</tr>
<tr>
<td>KVM Compute Node</td>
<td>nova-compute</td>
<td>service openstack-nova-compute status</td>
<td>service openstack-nova-compute start</td>
</tr>
<tr>
<td></td>
<td>nova-api-metadata</td>
<td>service openstack-nova-api-metadata status</td>
<td>service openstack-nova-api-metadata start</td>
</tr>
<tr>
<td></td>
<td>neutron-linuxbridge-agent</td>
<td>service neutron-linuxbridge-agent status</td>
<td>service neutron-linuxbridge-agent start</td>
</tr>
</tbody>
</table>

In a high availability topology, services on central server 2 and KVM/VMware region servers are configured highly available on respective secondary nodes. They are illustrated in the following table:

### Table 16. Servers and their high availability configuration

<table>
<thead>
<tr>
<th>Server name</th>
<th>Components deployed</th>
<th>Command to check the service</th>
<th>Command to start the service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Server 2 Primary</td>
<td>openstack-ceilometer-api</td>
<td>service openstack-ceilometer-api status</td>
<td>service openstack-ceilometer-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-keystone</td>
<td>service openstack-keystone status</td>
<td>service openstack-keystone start</td>
</tr>
<tr>
<td></td>
<td>httpd</td>
<td>service httpd status</td>
<td>service httpd start</td>
</tr>
<tr>
<td></td>
<td>ihs</td>
<td>service ihs status</td>
<td>service ihs start</td>
</tr>
<tr>
<td></td>
<td>bpm</td>
<td>service bpm status</td>
<td>service bpm start</td>
</tr>
<tr>
<td></td>
<td>scui</td>
<td>service scui status</td>
<td>service scui start</td>
</tr>
<tr>
<td></td>
<td>pcg</td>
<td>service pcg status</td>
<td>service pcg start</td>
</tr>
<tr>
<td>Central Server 2 Secondary</td>
<td>openstack-keystone</td>
<td>service openstack-keystone status</td>
<td>service openstack-keystone start</td>
</tr>
<tr>
<td></td>
<td>httpd</td>
<td>service httpd status</td>
<td>service httpd start</td>
</tr>
<tr>
<td></td>
<td>ihs</td>
<td>service ihs status</td>
<td>service ihs start</td>
</tr>
<tr>
<td></td>
<td>bpm</td>
<td>service bpm status</td>
<td>service bpm start</td>
</tr>
<tr>
<td></td>
<td>scui</td>
<td>service scui status</td>
<td>service scui start</td>
</tr>
<tr>
<td>Server name</td>
<td>Components deployed</td>
<td>Command to check the service</td>
<td>Command to start the service</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------</td>
<td>------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>KVM Region Server Primary</td>
<td>openstack-ceilometer-central</td>
<td>service openstack-ceilometer-central status</td>
<td>service openstack-ceilometer-central start</td>
</tr>
<tr>
<td></td>
<td>openstack-ceilometer-collector</td>
<td>service openstack-ceilometer-collector status</td>
<td>service openstack-ceilometer-collector start</td>
</tr>
<tr>
<td></td>
<td>openstack-ceilometer-compute</td>
<td>service openstack-ceilometer-compute status</td>
<td>service openstack-ceilometer-compute start</td>
</tr>
<tr>
<td></td>
<td>openstack-ceilometer-agent-notification</td>
<td>service openstack-ceilometer-agent-notification status</td>
<td>service openstack-ceilometer-agent-notification start</td>
</tr>
<tr>
<td></td>
<td>openstack-cinder-api</td>
<td>service openstack-cinder-api status</td>
<td>service openstack-cinder-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-cinder-scheduler</td>
<td>service openstack-cinder-scheduler status</td>
<td>service openstack-cinder-scheduler start</td>
</tr>
<tr>
<td></td>
<td>openstack-cinder-volume</td>
<td>service openstack-cinder-volume status</td>
<td>service openstack-cinder-volume start</td>
</tr>
<tr>
<td></td>
<td>openstack-glance-api</td>
<td>service openstack-glance-api status</td>
<td>service openstack-glance-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-glance-registry</td>
<td>service openstack-glance-registry status</td>
<td>service openstack-glance-registry start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-api</td>
<td>service openstack-heat-api status</td>
<td>service openstack-heat-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-api-cfn</td>
<td>service openstack-heat-api-cfn status</td>
<td>service openstack-heat-api-cfn start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-api-cloudwatch</td>
<td>service openstack-heat-api-cloudwatch status</td>
<td>service openstack-heat-api-cloudwatch start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-engine</td>
<td>service openstack-heat-engine status</td>
<td>service openstack-heat-engine start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-api</td>
<td>service openstack-nova-api status</td>
<td>service openstack-nova-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-cert</td>
<td>service openstack-nova-cert status</td>
<td>service openstack-nova-cert start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-conductor</td>
<td>service openstack-nova-conductor status</td>
<td>service openstack-nova-conductor start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-consoleauth</td>
<td>service openstack-nova-consoleauth status</td>
<td>service openstack-nova-consoleauth start</td>
</tr>
</tbody>
</table>
Table 16. Servers and their high availability configuration (continued)

<table>
<thead>
<tr>
<th>Server name</th>
<th>Components deployed</th>
<th>Command to check the service</th>
<th>Command to start the service</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KVM Region Server Primary (continued)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>openstack-nova-metadata-api</td>
<td>service openstack-nova-metadata-api status</td>
<td>service openstack-nova-metadata-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-novncproxy</td>
<td>service openstack-nova-novncproxy status</td>
<td>service openstack-nova-novncproxy start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-scheduler</td>
<td>service openstack-nova-scheduler status</td>
<td>service openstack-nova-scheduler start</td>
</tr>
<tr>
<td></td>
<td>neutron-server</td>
<td>service neutron-server status</td>
<td>service neutron-server start</td>
</tr>
<tr>
<td><strong>KVM Region Server Secondary</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>openstack-cinder-api</td>
<td>service openstack-cinder-api status</td>
<td>service openstack-cinder-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-cinder-scheduler</td>
<td>service openstack-cinder-scheduler status</td>
<td>service openstack-cinder-scheduler start</td>
</tr>
<tr>
<td></td>
<td>openstack-glance-registry</td>
<td>service openstack-glance-registry status</td>
<td>service openstack-glance-registry start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-api</td>
<td>service openstack-heat-api status</td>
<td>service openstack-heat-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-api</td>
<td>service openstack-nova-api status</td>
<td>service openstack-nova-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-cert</td>
<td>service openstack-nova-cert status</td>
<td>service openstack-nova-cert start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-conductor</td>
<td>service openstack-nova-conductor status</td>
<td>service openstack-nova-conductor start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-consoleauth</td>
<td>service openstack-nova-consoleauth status</td>
<td>service openstack-nova-consoleauth start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-metadata-api</td>
<td>service openstack-nova-metadata-api status</td>
<td>service openstack-nova-metadata-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-novncproxy</td>
<td>service openstack-nova-novncproxy status</td>
<td>service openstack-nova-novncproxy start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-scheduler</td>
<td>service openstack-nova-scheduler status</td>
<td>service openstack-nova-scheduler start</td>
</tr>
<tr>
<td></td>
<td>neutron-server</td>
<td>service neutron-server status</td>
<td>service neutron-server start</td>
</tr>
</tbody>
</table>
Table 16. Servers and their high availability configuration (continued)

<table>
<thead>
<tr>
<th>Server name</th>
<th>Components deployed</th>
<th>Command to check the service</th>
<th>Command to start the service</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMware Region Server Primary</td>
<td>openstack-ceilometer-central</td>
<td>service openstack-ceilometer-central status</td>
<td>service openstack-ceilometer-central start</td>
</tr>
<tr>
<td></td>
<td>openstack-ceilometer-collector</td>
<td>service openstack-ceilometer-collector status</td>
<td>service openstack-ceilometer-collector start</td>
</tr>
<tr>
<td></td>
<td>openstack-ceilometer-compute</td>
<td>service openstack-ceilometer-compute status</td>
<td>service openstack-ceilometer-compute start</td>
</tr>
<tr>
<td></td>
<td>openstack-cinder-api</td>
<td>service openstack-cinder-api status</td>
<td>service openstack-cinder-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-cinder-scheduler</td>
<td>service openstack-cinder-scheduler status</td>
<td>service openstack-cinder-scheduler start</td>
</tr>
<tr>
<td></td>
<td>openstack-cinder-volume</td>
<td>service openstack-cinder-volume status</td>
<td>service openstack-cinder-volume start</td>
</tr>
<tr>
<td></td>
<td>openstack-glance-api</td>
<td>service openstack-glance-api status</td>
<td>service openstack-glance-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-glance-registry</td>
<td>service openstack-glance-registry status</td>
<td>service openstack-glance-registry start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-api</td>
<td>service openstack-heat-api status</td>
<td>service openstack-heat-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-api-cfn</td>
<td>service openstack-heat-api-cfn status</td>
<td>service openstack-heat-api-cfn start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-api-cloudwatch</td>
<td>service openstack-heat-api-cloudwatch status</td>
<td>service openstack-heat-api-cloudwatch start</td>
</tr>
<tr>
<td></td>
<td>openstack-heat-engine</td>
<td>service openstack-heat-engine status</td>
<td>service openstack-heat-engine start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-api</td>
<td>service openstack-nova-api status</td>
<td>service openstack-nova-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-cert</td>
<td>service openstack-nova-cert status</td>
<td>service openstack-nova-cert start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-compute</td>
<td>service openstack-nova-compute status</td>
<td>service openstack-nova-compute start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-conductor</td>
<td>service openstack-nova-conductor status</td>
<td>service openstack-nova-conductor start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-consoleauth</td>
<td>service openstack-nova-consoleauth status</td>
<td>service openstack-nova-consoleauth start</td>
</tr>
</tbody>
</table>
Table 16. Servers and their high availability configuration (continued)

<table>
<thead>
<tr>
<th>Server name</th>
<th>Components deployed</th>
<th>Command to check the service</th>
<th>Command to start the service</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMware Region Server Primary (continued)</td>
<td>openstack-nova-metadata-api</td>
<td>service openstack-nova-metadata-api status</td>
<td>service openstack-nova-metadata-api start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-novncproxy</td>
<td>service openstack-nova-novncproxy status</td>
<td>service openstack-nova-novncproxy start</td>
</tr>
<tr>
<td></td>
<td>openstack-nova-scheduler</td>
<td>service openstack-nova-scheduler status</td>
<td>service openstack-nova-scheduler start</td>
</tr>
<tr>
<td></td>
<td>neutron-linuxbridge-agent</td>
<td>service neutron-linuxbridge-agent status</td>
<td>service neutron-linuxbridge-agent start</td>
</tr>
<tr>
<td></td>
<td>neutron-server</td>
<td>service neutron-server status</td>
<td>service neutron-server start</td>
</tr>
</tbody>
</table>
Table 16. Servers and their high availability configuration (continued)

<table>
<thead>
<tr>
<th>Server name</th>
<th>Components deployed</th>
<th>Command to check the service</th>
<th>Command to start the service</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMware Region Server</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secondary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>openstack-cinder-api</td>
<td>service</td>
<td>openstack-cinder-api status</td>
<td>service openstack-cinder-api start</td>
</tr>
<tr>
<td>openstack-cinder-scheduler</td>
<td>service</td>
<td>openstack-cinder-scheduler status</td>
<td>service openstack-cinder-scheduler start</td>
</tr>
<tr>
<td>openstack-glance-registry</td>
<td>service</td>
<td>openstack-glance-registry status</td>
<td>service openstack-glance-registry start</td>
</tr>
<tr>
<td>openstack-heat-api</td>
<td>service</td>
<td>openstack-heat-api status</td>
<td>service openstack-heat-api start</td>
</tr>
<tr>
<td>openstack-nova-api</td>
<td>service</td>
<td>openstack-nova-api status</td>
<td>service openstack-nova-api start</td>
</tr>
<tr>
<td>openstack-nova-cert</td>
<td>service</td>
<td>openstack-nova-cert status</td>
<td>service openstack-nova-cert start</td>
</tr>
<tr>
<td>openstack-nova-compute</td>
<td>service</td>
<td>openstack-nova-compute status</td>
<td>service openstack-nova-compute start</td>
</tr>
<tr>
<td>openstack-nova-conductor</td>
<td>service</td>
<td>openstack-nova-conductor status</td>
<td>service openstack-nova-conductor start</td>
</tr>
<tr>
<td>openstack-nova-consoleauth</td>
<td>service</td>
<td>openstack-nova-consoleauth status</td>
<td>service openstack-nova-consoleauth start</td>
</tr>
<tr>
<td>openstack-nova-metadata-api</td>
<td>service</td>
<td>openstack-nova-metadata-api status</td>
<td>service openstack-nova-metadata-api start</td>
</tr>
<tr>
<td>openstack-nova-novncproxy</td>
<td>service</td>
<td>openstack-nova-novncproxy status</td>
<td>service openstack-nova-novncproxy start</td>
</tr>
<tr>
<td>openstack-nova-scheduler</td>
<td>service</td>
<td>openstack-nova-scheduler status</td>
<td>service openstack-nova-scheduler start</td>
</tr>
<tr>
<td>neutron-linuxbridge-agent</td>
<td>service</td>
<td>neutron-linuxbridge-agent status</td>
<td>service neutron-linuxbridge-agent start</td>
</tr>
<tr>
<td>neutron-server</td>
<td>service</td>
<td>neutron-server status</td>
<td>service neutron-server start</td>
</tr>
</tbody>
</table>

Refer to the System automation policies section for information about the high availability policies of these services. If a service is configured active-active, it should be running on both the primary and the secondary nodes. If a service is configured active-passive, only one instance of the service should be running, on either the primary or secondary node.

Some services do not apply to the current configuration, therefore must not be running at all on either the primary or secondary node, for example openstack-nova-console, openstack-nova-xvpncproxy.
Restarting NoSQL

To restart NoSQL, perform the following procedure.

Procedure

1. Log in as DB2 instance user (for example, db2inst1) by running the following command:
   
   ```sh
   su - db2inst1
   ```

2. Create the logs directory and prepare the wplistener.properties file by running the following commands:
   
   ```sh
   mkdir -p ~/json/logs
   cat > ~/json/wplistener.properties
   # Adjust below variables according your environment
   # database name used by ceilometer
   dbName=OPENSTACK
   # mongo port used by ceilometer
   mongoPort=27017
   # Ceilometer database userid
   userid=ceil
   # Ceilometer database password
   password=passw0rd
   # Name of the Host running the noSQL process
   noSQLHost=localhost
   # Type Ctrl + D below to close the stream
   ```

3. Restart NoSQL by running the following commands:
   
   ```sh
   export JAVA_HOME=/sqllib/java/jdk64
   cd ~/sqllib/json/bin
   # stop noSQL
   ./wplistener.sh -logPath ~/json/logs -propsFile ~/json/wplistener.properties -shutdown
   # start noSQL. Make sure the according database has been started
   nohup ./wplistener.sh -logPath ~/json/logs -propsFile ~/json/wplistener.properties -start &> ~/json/logs/start.out &
   ```

Managing high availability

The Distributed Active-Active (High Availability) installation of IBM Cloud Orchestrator uses System Automation Application Manager and System Automation for Multiplatforms to achieve high availability.

Learn about System Automation Application Manager and System Automation for Multiplatforms, how to configure these products for IBM Cloud Orchestrator, and how to use them to start and stop the IBM Cloud Orchestrator management stack.

High-availability solutions

Learn how to set up an IBM Cloud Orchestrator management stack with high-availability quality of service (QoS), and reduce the downtime of the IBM Cloud Orchestrator management stack.

IBM Cloud Orchestrator achieves high availability by using redundant active-active installed components, where possible.

The high-availability capabilities of the IBM Cloud Orchestrator management stack are supported by using the following products:
System Automation Application Manager
System Automation Application Manager automates the availability of resources by starting and stopping resources automatically and in the correct sequence. System Automation Application Manager uses agentless adapters to monitor and control remote applications. System Automation Application Manager ensures availability and provides automation of these services across operating-system boundaries. System Automation Application Manager provides a centralized server that monitors and automatically stops, starts, or restarts the various applications on the virtual machines in the IBM Cloud Orchestrator environment.

System Automation for Multiplatforms
System Automation for Multiplatforms is a clustering solution that provides high-availability and automation features for critical components such as applications, network interfaces, virtual IP addresses, and storage. In the IBM Cloud Orchestrator environment, System Automation for Multiplatforms can monitor and fail over critical components that rely on an active-standby setup. System Automation for Multiplatforms uses automation capabilities to ensure that actions are completed in the correct order if a component fails. System Automation for Multiplatforms automates services on Central Server 2, on KVM Region Servers, and on VMware Region Servers.

These high-availability solutions are compatible and can be used together to achieve high availability of the IBM Cloud Orchestrator management stack. When you use System Automation Application Manager and System Automation for Multiplatforms with IBM Cloud Orchestrator, you can recover the IBM Cloud Orchestrator environment after hardware or software failure. This solution reduces the downtime of the IBM Cloud Orchestrator management stack. In addition, operation of the IBM Cloud Orchestrator management stack is simplified.

To use the hypervisor high-availability capabilities, you must complete extra manual steps at installation time.

For information about installing high availability, see "Deploying the Distributed Active-Active (High Availability) topology” on page 69.

High-availability configurations
For the IBM Cloud Orchestrator management stack, high-availability quality of service (QoS) can be provided in an active-active configuration or in an active-standby configuration.

In a high-availability setup, the applications and services are usually configured as redundant. One of the main differences between the high-availability configurations is how the second server is set up.

active-active
In an active-active setup, the application is running on all instances. A load balancer or proxy can be deployed to accept requests and to distribute the requests to the nodes. Balancing the load in this way enhances performance because more systems can handle the workload simultaneously. To use the active-active setup, an application usually must be designed to provide this capability: for example, to store and access the underlying data.

active-standby
In an active-standby setup, the application is running on only one instance at a time. It is often necessary to enforce this limit because the shared
storage or database might become corrupted if accessed by multiple instances simultaneously. Applications that do not have an active-active design are usually run in the active-standby mode when they are made highly available.

System Automation for Multiplatforms provides services to both types of high-availability configurations:

- For components in an active-standby setup, which is also known as warm standby, System Automation for Multiplatforms provides monitoring and failover capabilities, and ensures that exactly one instance of the service is running at any time.

  The application or service is installed and readily configured on both nodes. If data access is necessary, the data can be made available to both nodes. If an unrecoverable error occurs on the first node, the application and all related services are stopped and automatically restarted on the second node.

- For components in an active-active setup, System Automation for Multiplatforms provides monitoring and restart capabilities, and ensures that the service remains online even if temporary outages occur.

  The application or service is installed and readily configured in the active-active setup on both nodes. If an unrecoverable error occurs on any node, System Automation for Multiplatforms detects the problem and automatically recovers, which ensures maximum performance.

**Goal-driven automation**

Goal-driven automation helps to maintain each component in the desired state, when IBM Cloud Orchestrator is configured as highly available. Do not interfere with goal-driven automation by doing manual actions.

Automation can be goal-driven or command-driven. In command-driven automation, the command is issued without any preconditions. In goal-driven automation, the desired state is computed from multiple persistent inputs.

When IBM Cloud Orchestrator is configured as highly available, the components are managed by System Automation Application Manager and System Automation for Multiplatforms. Both of these system automation products try to maintain a desired state for every resource that they manage. This desired state can be specified as *online* or *offline*, and is calculated from the following input factors:

- The default state that is defined in the policy
- Operator requests sent from the System Automation Application Manager and System Automation for Multiplatforms user interfaces
- Relationship to other resources, which might require a component to start or stop

Therefore, it is important that you manage these resources by changing their desired state. If you start or stop resources outside the scope of System Automation Application Manager or System Automation for Multiplatforms, the action is evaluated as an unwanted deviation from the automation goal, and can result in countermeasures.

For more information about goal-driven automation and how to manage resources, see the following web pages:

- [Goal-driven automation](http://www-01.ibm.com/support/knowledgecenter/api/content/SSPQ7D_4.1.0/com.ibm.saam.doc_4.1/totechov_autogoalrequest.html?locale=en)
Using System Automation Application Manager

Use System Automation Application Manager to automate the availability of resources. System Automation Application Manager starts and stops resources automatically and in the correct sequence.

System Automation Application Manager can detect if a software component of IBM Cloud Orchestrator failed, and can then automatically restart it. As a result, you can achieve an improved availability solution for the IBM Cloud Orchestrator management stack. System Automation Application Manager continuously monitors all the components and subcomponents of the IBM Cloud Orchestrator management stack. In case of failure, it reacts by automatically restarting the failed components and the related subcomponents. The System Automation Application Manager configuration includes automation policies that define the start and stop dependency graph for all the IBM Cloud Orchestrator components. As a result, the operating team can start, stop, and monitor the state of the IBM Cloud Orchestrator management stack. System Automation Application Manager also simplifies the operating tasks because it is possible to control the IBM Cloud Orchestrator management stack in a more granular manner on the component level. For example, with a single request you can recycle one component such as Business Process Manager.

During installation, the saamuser user is created on the system where System Automation Application Manager is installed. The saamuser user is also created on all systems that are accessed by the System Automation Application Manager agentless adapter component. This user is authorized to control the services that are managed by System Automation Application Manager. This authorization is granted by a user-specific configuration in the /etc/sudoers.d/saam file.

Using System Automation for Multiplatforms

Learn about System Automation for Multiplatforms resources, ServiceIPs, and relationships.

The following concepts are used by System Automation for Multiplatforms:

Resources

To provide failover capabilities, resources in System Automation for Multiplatforms are modelled as floater resources. A resource that can be started on two nodes is visible as three resources: the overall resource container (also called the aggregate resource), and a representation of the resource on each node (also called constituent resources).

Modelling a resource in this way results in the following automation behaviour:

- A floater resource ensures that only one constituent resource is started at any time. If more than one active constituent is detected, the additional constituents are stopped immediately.
A floater resource has a desired state. If the resource is not in the desired state, System Automation for Multiplatforms initiates actions to achieve the desired state. A resource that fails is restarted in place.

If a resource experiences nonrecoverable errors, System Automation for Multiplatforms attempts a failover: the constituent that is experiencing the errors is marked as failed, and the other constituent is started.

System Automation for Multiplatforms recognizes only status changes made through its own interfaces as changes to the desired state. If operations occur on a resource, System Automation for Multiplatforms interprets those changes as unwanted changes and reestablishes the desired state.

ServiceIPs
A typical problem in highly available services is reaching those services. Independent access to the service must be established, regardless of which constituent is currently online. To provide this access, System Automation for Multiplatforms enables you to define a ServiceIP. A ServiceIP is a type of floater resource and has the same properties as other floater resources. Instead of managing an application, the ServiceIP provides an alias to a network interface: for example, eth0:0 for eth0, with a new IP address. ServiceIPs are also often called virtual IPs.

Relationships
A strong tie exists between an application and the IP address where the application is available. If you automate an application and a ServiceIP, you must ensure that both run on the same server, and that the ServiceIP is available for the application at all times. To model such relations, System Automation for Multiplatforms can define relationships of various types between resources. The basic relationships are the start-stop dependencies (for example, the ServiceIP must always be online before the HTTP Server is started) and the location dependencies (for example, the ServiceIP and the HTTP Server must always be on the same server).

The relationship used most commonly in the IBM Cloud Orchestrator high-availability topology is the dependsOn relationship. The dependsOn relationship combines multiple relationships and provides the following automation behaviour for two resources where Resource A dependsOn Resource B:

1. Before Resource A is started, Resource B must be started. Resource A waits for Resource B to be online before starting.
2. Before Resource B is stopped, Resource A must be stopped. Resource B waits for Resource A to be offline before stopping.
3. Resource A must always be started on a server where Resource B is already online.
4. If Resource B fails, Resource A is stopped. Then Rule #1 applies again.
System automation policies

When you install IBM Cloud Orchestrator in a high-availability topology, the required automation policies are created automatically during installation.

Automation policies are XML files that contain a description of the automated resources. The policy files include the following information:

- The systems that are run by the component
- The commands that are used to start, stop, and monitor the system
- The timeouts that should be enforced on these resources
- The relationships that should be considered when managing those resources

The following policies are used in IBM Cloud Orchestrator:

- One policy for every System Automation for Multiplatforms cluster
- One policy for every Agentless Adapter domain
- One end-to-end policy for the System Automation Application Manager domain which integrates all other policies with one another

When you install IBM Cloud Orchestrator in a high-availability topology, the following required automation policies are created automatically during the installation:

- The System Automation for Multiplatforms policies are created and automatically activated on the nodes of the System Automation for Multiplatforms cluster.
- The System Automation Application Manager Agentless Adapter policies are created on the node where System Automation Application Manager is installed, and are activated from there. Exception: You must manually configure the external IBM DB2 server as described in “Configuring the external database server for the high-availability topology” on page 145.
- The end-to-end policy for the System Automation Application Manager must be adjusted for every deployment of a Region Server. The installer automatically extends the end-to-end policy file on the System Automation Application Manager server, so that it can be activated manually.
- All of the above resources are combined in a System Automation Application Manager end-to-end automation policy. This policy defines the Cloud Orchestrator group. This group can be used to monitor the health of the IBM Cloud Orchestrator system, and to start and stop IBM Cloud Orchestrator as a single entity.

Central Server 2 automation policy

When IBM Cloud Orchestrator is configured as highly available, a system automation policy is created for Central Server 2.

In the high-availability topology, Central Server 2 is configured as a redundant System Automation for Multiplatforms cluster consisting of two nodes. Some components are configured as active-active, and some components are configured as active-passive. System Automation for Multiplatforms also manages several relationships between the components to ensure that the components are started and stopped in the correct sequence. The following resources are managed in the Central Server 2 cluster:
Table 17. Resources managed by the Central Server 2 automation policy

<table>
<thead>
<tr>
<th>Component</th>
<th>Resource name in policy</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM HTTP Server</td>
<td>ihs</td>
<td>Active-passive</td>
</tr>
<tr>
<td>haproxy</td>
<td>haproxy</td>
<td>Active-passive</td>
</tr>
<tr>
<td>IBM Business Process Manager</td>
<td>bpm</td>
<td>Active-active</td>
</tr>
<tr>
<td>Keystone</td>
<td>keystone</td>
<td>Active-active</td>
</tr>
<tr>
<td>Horizon</td>
<td>horizon</td>
<td>Active-active</td>
</tr>
<tr>
<td>Public Cloud Gateway</td>
<td>pcg</td>
<td>Active-active</td>
</tr>
<tr>
<td>Self-service user interface</td>
<td>scui</td>
<td>Active-active</td>
</tr>
</tbody>
</table>

The Central Server 2 automation policy is configured as follows:

- Active-active resources run on both nodes. If a resource fails on one node, System Automation for Multiplatforms restarts the resource on that node. If at least one resource is still running, the service is monitored as online.
- Active-passive resources run on one node, preferably the first node. If the online resource fails, System Automation for Multiplatforms restarts the resource on that node, if possible. If a permanent error is detected, for example, the start script returns an error code, a failover to the second node is initiated.
- IBM HTTP Server and haproxy depend on an IBM.ServiceIP resource that represents a virtual IP address. If IBM HTTP Server or haproxy fails over, the ServiceIP fails over first. All clients of these two services must connect to the ServiceIP at all times. Otherwise, such clients cannot connect to the service (IBM HTTP Server or haproxy) when it fails over.

**Note:** IBM HTTP Server and haproxy share the same ServiceIP. Therefore, both services always run on the same node.

Region Server automation policy

When IBM Cloud Orchestrator is configured as highly available, a system automation policy is created for each Region Server.

In the high-availability topology, each Region Server is configured as a redundant System Automation for Multiplatforms cluster consisting of two nodes. Some components are configured as active-active, and some components are configured as active-passive. System Automation for Multiplatforms also manages several relationships between the components to ensure that the components are started and stopped in the correct sequence. The following resources are managed in the Region Server clusters:

Table 18. Resources managed by the Region Server automation policy

<table>
<thead>
<tr>
<th>Component</th>
<th>Resource name in policy</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>haproxy</td>
<td>haproxy</td>
<td>Active-passive</td>
</tr>
<tr>
<td>qpid</td>
<td>qpid</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>qpid-primary</td>
<td>Active-passive with hot standby</td>
</tr>
<tr>
<td>Ceilometer</td>
<td>ceilometer-central-agent</td>
<td>Single Instance</td>
</tr>
<tr>
<td></td>
<td>ceilometer-collector</td>
<td>Single Instance</td>
</tr>
<tr>
<td></td>
<td>ceilometer-compute-agents</td>
<td>Single Instance</td>
</tr>
</tbody>
</table>
Table 18. (continued). Resources managed by the Region Server automation policy

<table>
<thead>
<tr>
<th>Component</th>
<th>Resource name in policy</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cinder</td>
<td>cinder-api</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>cinder-scheduler</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>cinder-volume</td>
<td>Single Instance</td>
</tr>
<tr>
<td>Glance</td>
<td>glance-api</td>
<td>Single Instance</td>
</tr>
<tr>
<td></td>
<td>glance-registry</td>
<td>Active-active</td>
</tr>
<tr>
<td>Heat</td>
<td>heat-api-cfn</td>
<td>Single Instance</td>
</tr>
<tr>
<td></td>
<td>heat-api-cloudwatch</td>
<td>Single Instance</td>
</tr>
<tr>
<td></td>
<td>heat-api</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>heat-engine</td>
<td>Single Instance</td>
</tr>
<tr>
<td>Neutron</td>
<td>neutron-api</td>
<td>Active-active</td>
</tr>
<tr>
<td>Nova</td>
<td>nova-api</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>nova-cells</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>nova-cert</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>nova-compute-vmwarapi</td>
<td>Single Instance</td>
</tr>
<tr>
<td></td>
<td>nova-conductor</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>nova-consoleauth</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>nova-novncproxy</td>
<td>Active-active</td>
</tr>
<tr>
<td></td>
<td>nova-scheduler</td>
<td>Active-active</td>
</tr>
</tbody>
</table>

The Region Server automation policy is configured as follows:

- Single-instance resources run on the primary node only. If this resource fails, System Automation for Multiplatforms restarts the resource on the primary node. If the resource encounters a non-recoverable error, System Automation for Multiplatforms indicates the error, but the resource remains in the failed state.

- Active-active resources run on both nodes. If a resource fails on one node, System Automation for Multiplatforms restarts the resource on that node. If at least one resource is still running, the service is monitored as online.

- Active-passive resources run on one node, preferably the primary node. If the online resource fails, System Automation for Multiplatforms restarts the resource on that node, if possible. If a permanent error is detected, for example, the start script returns an error code, a failover to the second node is initiated.

- qpid is handled by two resources: qpid controls the qpid daemon, and qpid-primary controls the assignment of the primary role and secondary role to the resources. The qpid resource is active/active and the qpid-primary resource is active-passive. The daemon resources are online on both nodes, but only the primary node (Node A) accepts requests. The secondary resource acts as a replication server, and provides a backup of the message queue: the message broker copies all messages to the secondary node (Node B) before sending the acknowledgment of receipt. When a failover occurs, the daemon resource is restarted to ensure that the roles switch correctly: the original secondary node (Node B) now becomes the primary node, and the qpid daemon runs on this node. When the original primary node (Node A) comes back online, it starts as the secondary node: the qpid-primary resource now runs on this node. After the nodes reconnect to each other, the message queue is backed up on the new secondary node (Node A).
qpid-primary and haproxy depend on an IBM.ServiceIP resource that represents a virtual IP address. If qpid-primary or haproxy fails over, the ServiceIP fails over first. All clients of these two services must connect to the ServiceIP at all times. Otherwise, such clients cannot connect to the service (qpid-primary or haproxy) when it fails over.

Note: qpid-primary and haproxy share the same ServiceIP. Therefore, both services always run on the same node. The qpid resource always runs on both nodes and is independent of the IP address.

Controlling the management stack
Use System Automation Application Manager to control the IBM Cloud Orchestrator management stack to shut down the whole stack or specific services for maintenance and planned outages.

About this task
In addition to stopping a specific service, System Automation Application Manager can also ensure that any dependencies between services are automatically resolved. For example, if you want to stop DB2 for a backup, before stopping the DB2 service, all the services requiring DB2 are automatically stopped.

Important: If you plan to reboot or shut down an IBM Cloud Orchestrator virtual machine, you must stop the IBM Cloud Orchestrator services on this virtual machine via System Automation Application Manager before the OS is shut down or rebooted. Otherwise, System Automation Application Manager might set a resource in error. To resolve the error, follow the steps described in “Troubleshooting a high-availability installation” on page 837. After the virtual machine is started again, you can start the services by cancelling the offline request as described in the following procedure.

To stop an IBM Cloud Orchestrator service by using System Automation Application Manager, perform the following steps:

Procedure
1. Log on to System Automation Application Manager web user interface at https://<SAAM server hostname>:16311/ibm/console/. Specify the user and password that you defined during the installation. The default user is eezaadmin.
2. Click the Operate end-to-end resources link. A view of the managed end to end resources is displayed.
3. Click the plus sign in front of the top level resource SCOsaaam to expand and show the lower level resources.
4. In the Resources section for the policy, click the SCO resource group to expand the list of all the related resources and resource groups.
5. If you want to stop the whole IBM Cloud Orchestrator management stack, select the top level resource SCOsaaam and right-click the resource. Select Request Offline. Enter a comment in the displayed window and click Submit.
6. If you want to stop a specific resource, right-click the resource name to select it and click Request Offline... Enter a comment in the displayed window and click Submit. The service and all the dependent services are stopped. This only works for resources that are managed by the agentless adapter.
7. To restart a resource or a resource group, right-click on the resource and select Cancel Request.
**Results**

You stopped and restarted the IBM Cloud Orchestrator services.

For more information about using System Automation Application Manager, see the System Automation Application Manager, Administrator's and User's Guide.

---

**Managing settings**

You can configure product settings before building a cloud infrastructure.

**Before you begin**

You must be assigned the admin role to manage the product settings.

**Managing email delivery**

The mail delivery function is used to send event notifications.

**Before you begin**

You must be assigned the admin role to perform these steps.

**About this task**

The mail delivery function is used to automatically notify the owner of a virtual system instance, a virtual system pattern, or a virtual image about the following events:

- Virtual system instance is created.
- Virtual system instance has successfully started.
- Virtual system instance failed to start successfully.
- Virtual system instance is going to be removed due to expired reservation.
- Virtual system pattern is created.
- Virtual system pattern is going to be removed due to expired reservation.
- Virtual image is exported.
- Virtual image is imported.

**Note:** For vsys.next pattern, only the mail of failure deployment can be received, the mail of deployment start and successful deployment can't be received.

To configure the mail delivery function, specify the required Simple Mail Transfer Protocol (SMTP) server to be used for IBM Cloud Orchestrator by performing the following steps:

**Procedure**

1. Click PATTERNS > Deployer Administration > Settings.
2. Expand Mail Delivery.
3. Add an SMTP server. Provide the IP address or host name for the SMTP server to be used for IBM Cloud Orchestrator.
4. Add a reply-to address. The email address for the administrator should be used for this field.
Results

The mail deliver function has been configured to send event notifications.

Configuring OpenStack synchronization

You can optionally change the default synchronization interval with the OpenStack environment.

About this task

In IBM Cloud Orchestrator, the lists of cloud groups, hypervisors, and IP groups are automatically imported from the OpenStack environment. This information is periodically refreshed so that each change you make in the OpenStack environment is automatically applied also in the IBM Cloud Orchestrator environment.

To change the synchronization time interval, edit the /opt/ibm/rainmaker/purescale.app/private/expanded/ibm/scp.ui-1.0.0/config/openstack.config file (on the Central Server 3) and change the following statement:

```bash
/config/openstack = {
...

    "synchronization": {
        "initial_delay_sec": 60,
        "interval_sec": 600,
        "hypervisor_discovery_timeout_sec": 600,
        "hypervisor_discovery_check_interval_sec": 10
    }
}
```

where

- **initial_delay**
  Specifies the time interval (in seconds) between the start of the Workload Deployer server and the first synchronization with the OpenStack environment. It is optional. The default value is 60.

- **interval_sec**
  Specifies the time interval (in seconds) between the synchronizations with the OpenStack environment. The default value is 600.

- **hypervisor_discovery_timeout_sec**
  Specifies the timeout after which the operation will be reported as failed.

- **hypervisor_discovery_check_interval_sec**
  Specifies the time interval for the hypervisor discovery operation. By default, Workload Deployer will check every 10 seconds for a maximum of 10 minutes.
Customizing the user interface

You can add views to the user interface in IBM Cloud Orchestrator as well as change the branding for each domain.

The user interface is updated by editing the customization metadata properties in the customizations.json file. A copy of this file is available in the following directory after IBM Cloud Orchestrator is installed: {server_location}/etc/customizations/template.

Note: {server location} is usually found in Central Server 2, path /opt/ibm/ccs/scui/. The path is the same on all topologies. This customizations.json file can be copied, updated, and added to a specific domain directory under the main customization directory so the UI can be customized. For example: {server_location}/etc/customizations/{domainName}.

Note: The template directory is present so that it can be duplicated and used as an example when you are creating the customizations for other domains.

The customizations.json file that is located in {server_location}/etc/customizations/default is used for customizations to the default domain. The default domain is available by default after installation in the following directory: {server_location}/etc/customizations.

The following topics explain the customizations.json file, the images folders and how they can be edited to update the user interface that is based on the domain to which you belong:

Branding the user interface per domain
You can change the domain branding and customize the user interface.

Metadata file properties in customization file:

The metadata file properties are contained in the customizations.json file. This file contains both content and style metadata properties that are used to manipulate the appearance of the IBM Cloud Orchestrator user interfaces for a particular domain.

The following section explains the metadata file properties:

Content properties
The content properties section of the file contains all the label and static type content that is customized. This content is generally text values and links to replacement images, for example, if you want to use a different logo. The following table shows the values that are contained in the customizations.json file:

<table>
<thead>
<tr>
<th>Content property values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text values</td>
<td>Items such as the title used in the main page banner and the product title that is used on the login page.</td>
</tr>
<tr>
<td>Image values</td>
<td>Items such as the product logo in the main page banner and the product logo that is used on the login page.</td>
</tr>
</tbody>
</table>
Table 19. (continued)

<table>
<thead>
<tr>
<th>Content property values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content values</td>
<td>Items such as words, sentences, or resource location paths. Content values substitute dynamically replaceable elements in the html template files in IBM Cloud Orchestrator</td>
</tr>
</tbody>
</table>

Style properties

The style properties section of the file contains all the css style classes, attributes, and elements that are customized. The style properties that are defined must be legal css because it is this metadata that forms the direct input for creating the css properties in the catalog-branding.css file. The catalog-branding.css file is dynamically rendered through the Common Custom Service. The catalog-branding.css is available at the following URL:

<scuiHostName:port>/style/catalog-branding.css

If you are logged in to a specific domain, the catalog-branding.css that is used is the one that is specified by the domains customization. If no catalog-branding.css is defined, the default css is used. When all of the properties are defined, the IBM Cloud Orchestrator user interface uses the Common Custom Service to change the look, feel, and style. The style values include items such as the background color used in the main page banner, the font, the logo, and so on.

The following example shows what the customizations.json metadata file looks like:

```
{
  "bannerLogo":="/customizations/template/images/IBM_logo.png",
  "bannerLogoAlt":"IBM Cloud Orchestrator Test",
  "title":"IBM Cloud Orchestrator Test",
  "backgroundColor": "teal",
  "bodyBackgroundColor": "beige",
  "bodyBackgroundImage": "",
  "loginLogo": "static/images/black-ibm-logo.png",
  "loginLogoAlt": "IBM Cloud Orchestrator Template Login Logo Alt Text"
}
```

IBM Cloud Orchestrator supports customizable properties that are defined in the customizations.json file. These supported customizable properties are described in the following table.

Table 20. Customizable properties

<table>
<thead>
<tr>
<th>Metadata property values</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>bannerLogo</td>
<td>File system path</td>
</tr>
<tr>
<td>bannerLogoAlt</td>
<td>Text</td>
</tr>
<tr>
<td>title</td>
<td>Text</td>
</tr>
<tr>
<td>bodyBackgroundColor</td>
<td>Text</td>
</tr>
<tr>
<td>bodyBackgroundImage</td>
<td>File system path</td>
</tr>
<tr>
<td>loginLogo</td>
<td>File system path</td>
</tr>
<tr>
<td>loginLogoAlt</td>
<td>Text</td>
</tr>
</tbody>
</table>
Note: Although you can enter an unlimited number of characters for the **title** property, only titles with less than 50 characters are displayed on the User Interface.

Example 1: The following customizations.json file shows an example with the title and logo in the banner changed and a new banner color with a gradient.

```json
{
  "bannerLogo": "/customization/images/Mylogo.png",
  "bannerLogoAlt": "Example Orchestrator",
  "title": "Example Orchestrator",
  "bannerBackgroundColor": "teal",
}
```

Example 2: The following customizations.json file is the same as Example 1 except that the background is set as an image and the banner color is solid blue.

```json
{
  "bannerLogo": "/customization/images/Mylogo.png",
  "bannerLogoAlt": "Example Orchestrator",
  "bannerTitle": "Example Orchestrator",
  "bannerBackgroundColor": "teal",
  "bodyBackgroundColor": "beige",
  "bodyBackgroundColorImage": "/customization/images/MyBackground.png"
}
```

**Note:** For the **bannerLogo** and **bodyBackgroundImage** properties, the image name specified must be preceded by /customization/images/. For example, if exampleLogo.png is the image file to use, then the property value is /customization/images/exampleLogo.png. The file in the customizations/{domainName}/images directory will be used for that property.

All base customization files that are used in the IBM Cloud Orchestrator user interface are found in the following location: {server_location}/etc/customizations/template. For domains that are not customized, the default content that is found in {server_location}/etc/customizations/default is used to alter the style and provide the content to the user interface.

**Note:** This {server_location}/etc/customizations/template directory must not be removed, as it is used as an example for all other customizations.

**How to customize the user interface:**

You can update the IBM Cloud Orchestrator user interface for a given domain.

**Procedure**

1. Create a directory under {server_location}/etc/customizations with the name of the domain that you want to customize. The customization metadata is stored in the customizations.json file in a specific domain directory under the main custom directory. For example: {server_location}/etc/customizations/{Domain1}

   **Note:** {server location} is usually found in Central Server 2, path /opt/ibm/ccs/scui/. The path is the same on all topologies.

   **Note:** Branding changes are only visible to the domain they are declared for, except the Default domain.

2. Create a file in the directory that you specified in step 1 and call it customizations.json.
Note: Copy the customizations.json file from the {server_location}/etc/customizations/template directory to the new domain, for example, Domain1 and edit the file instead of manually creating the file.

3. Create a folder that is called images under the customizations/{domainName} directory.

4. Copy the image file that is referenced in the content section of the customizations.json file into the same customizations/{domainName}/image directory. This method is used to ensure that all domain-specific customization content is in the correct location.

Note: You can also add and create new images and add them to the images folder. However, you must update the customizations.json file accordingly, based on any new image file that you add.

5. Restart the server and clear the browser cache. Alternatively, enter the following link in your browser to clear the server cache without having to restart the server: <scuiHostName:port>/customization/clearcustomcache

6. Log in to the IBM Cloud Orchestrator user interface with the domain that was customized and view the results to ensure that you are satisfied with the style updates.

Note: If an error is displayed in the user interface, it indicates that there is a problem with the customization. Complete the following steps:
- View the server logs to find more information about the error:
  /var/log/scoui.log and /var/log/scoui.trc.
- Ensure that the domain directory is correctly named, the customizations.json file is in the correct format, and its contents are correct.

Dashboard extensions

A dashboard extension file is a single html file that displays one or more BPM coaches. For the purpose of the dashboard extension these coaches should contain dashboards, graphs or other reporting elements developed in BPM.

The user defined extension html file contains a fragment of html without a <head> or a <body> element. These are included in the parent html file which embeds the supplied extension content.

The html file must contain <iframe> tags to specify the BPM dashboard coaches that you want to display on the page. In this case an <iframe> is used to embed one html document, the dashboard coach, within another html document, the parent extension. Define the width and height of the in-line coach frames to achieve the layout you want.

The following snippet of code is an example of a user defined html fragment with a BPM dashboard <iframe> that is contained in a sample extension html file:

```html
<div align="center">
  <iframe id="ifm" name="ifm" width="1000" height="1000" frameborder="0"
    src="{{bpmEndpoint}}/teamworks/process.lsw?
      zWorkFlowState=5&zProcessRef=/1.fc983f33-e98f-4999-b0b5-
      bd1e39d6102e&zBaseContext=2064.abec322d-430c-43dd-820a-
      9b223d29f9c3a:33127541:1461a68c1cd:-7ffe&applicationInstanceId=guid:6416d37cd5f92c3a:33127541:1461a68c1cd:-7ffe&applicationId=2"
    scrolling="auto" align="middle">
  </iframe>
</div>
```
In the example there is the following mustache variable defined \texttt{\{bpmEndpoint\}}. This utility variable is supplied by the extension framework. It enables extension content providers to locate and specify the base BPM host server url and port of an installed SCO instance in a generic way. Having this variable available to extension deployers means that no manual editing needs to be made to the html file after deployment to specify the location of the BPM server. For more information related to mustache see: \url{http://mustache.github.io/mustache.5.html}

Single sign-on is enabled between the Self Service UI and BPM, so displaying BPM coaches in the UI does not require any additional authentication configuration.

\textbf{Role based directory structure:}

Dashboards are role based. This allows service designers to have their dashboard extension content available to specific roles.

The directory structure that the extension content is deployed into is based on the SCO roles in the following table. The four out of the box SCO roles which the extension is compatible with are:

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
Name of role directory & Role \\
\hline
member & End User \\
admin & Cloud Administrator \\
domain_admin & Domain Administrator \\
catalogeditor & Service Designer \\
\hline
\end{tabular}
\end{table}

The dashboard folder is the parent extension directory. It contains a folder for each of the four roles. It is located on the icoui server at the following location:

{server_location}/etc/dashboard

e.g. \{server_location\}/etc/dashboard/admin or \{server_location\}/etc/dashboard/member

When a service designer wants to make a new dashboard available to an admin user, for example, they add the extension html file to the: \{server_location\}/etc/dashboard/admin directory

\textbf{Navigation elements and extension naming conventions:}

Naming extension files is important as these names drive the names of the navigation elements where the dashboards are accessed from.

Navigation elements should be meaningful so that you can easily locate important content in the UI. As these dashboard extension navigation elements are driven from the names of the dashboard extension files, these file names should be meaningful. Dashboard extensions appear as submenu items under the DASHBOARD menu and take the name of the extension file as their menu label. The label is the complete file name with any ordinal position specified and the file extension removed.

The following table describes an example of file names and their respective menu labels:
Table 22.

<table>
<thead>
<tr>
<th>File</th>
<th>Menu label</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 - Admin Extension Example.html</td>
<td>Admin Extension Example</td>
</tr>
<tr>
<td>02 - Member Extension Example.html</td>
<td>Member Extension Example</td>
</tr>
<tr>
<td>Network Dashboard.html</td>
<td>Network Dashboard</td>
</tr>
<tr>
<td>Performance Dashboard.html</td>
<td>Performance Dashboard</td>
</tr>
</tbody>
</table>

Ordinal numbering of extension files is included so that you can control the order in which the extension labels appear in the DASHBOARD menu. The ordinal numbering format convention is a number, then a hyphen (-), then the file name. Any file name starting with this pattern is placed in this relative numbered position in the sub menu. The pattern is stripped from the file name when constructing the label. If an ordinal numbering format convention is not used when naming extension files the files is added alphabetically.

Packaging and deployment:

A dashboard extension made available via the marketplace must be packaged as a compressed zip or tar.gz file.

The package structure must match the role based directory structure mentioned in the previous section. For example:

```plaintext
extensionExample.zip
  + dashboard
    + admin
      + 01 - My Admin Extension Example.html
```

To deploy the extension, extract the compressed file to `{server_location}/etc`. To have the new extension content appear in the UI, restart the icoui server. Alternatively, enter the following link in your browser to clear the navigation cache:

`<scuiHostName:port>/dashboardextension/clearcache`

**Note:** Before deploying a dashboard extension ensure that there are no extension files that have already been deployed previously in the extension directories with the same name. Any deployed file with the same name as one file contained in the package will get overwritten otherwise.

Managing security

You can manage users and the level of access that each user has in the IBM Cloud Orchestrator environment. You can assign which roles a user has on a specific project, as well as the primary project for a user. A user can have different roles on different projects. Both users and projects belong to a domain.

About this task

The OpenStack Compute system is designed to be used by many different cloud-computing consumers or customers, basically projects on a shared system, using role-based access assignments. Roles control the actions that a user is allowed to perform. For example, you can define that a user cannot allocate a public IP without the **admin** role. A user's access to particular images is limited by project, but the username and password are assigned per user. Key pairs granting access to an instance are enabled per user, but quotas to control resource consumption across available hardware resources are per project.
A project is an isolated resource container forming the principal organizational structure within the OpenStack environment. It consists of a separate VLAN, volumes, instances, images, keys, and users. For more information about OpenStack users and projects, see the OpenStack documentation.

For projects, quota controls are available to limit the following resources:
- Number of volumes that can be created
- Total size of all volumes within a project as measured in GB
- Number of instances that can be launched
- Number of processor cores that can be allocated
- Publicly accessible IP addresses

For more information about the OpenStack commands to be used to manage quota values in the projects, see “Configuring project quotas” on page 225.

When you create users and projects in IBM Cloud Orchestrator, they are actually being created in the underlying OpenStack environment. The roles that are defined in the OpenStack environment are used in IBM Cloud Orchestrator as described in “User roles in IBM Cloud Orchestrator” on page 206.

In IBM Cloud Orchestrator, the OpenStack concept of domain is also supported. A domain represents a customer (that is, for example an organization or a division) and the related resources as a segregated entity. Only users within that domain have access to these resources. In this way service providers can use the same IBM Cloud Orchestrator infrastructure to serve multiple customers. A domain can also be seen as a sort of container for projects and users. A user or a project can belong only to one domain. Domain administrators are authorized to create, update and delete resources related to the domain.

Important: IBM Cloud Orchestrator does not support the definition of same resource names in multiple domains at the same time (for example, a project with same name exists in different domains). The cloud administrator must ensure that there are no duplicated user names, project names, and domains.

When a user logs in to the IBM Cloud Orchestrator user interface, the user must specify the domain scope to which the user wants to be authenticated. In a single domain environment or if the user does not specify a domain, the users is authenticated to the Default domain. In addition to the domain-based authentication, when logging in, by default, the user is authenticated to scope of the primary project that was specified when the user was created. After successful authentication, the user can switch the project from the project list in the top banner of the user interface.

Important: A user always authenticates on behalf of its domain. Therefore, as a user, you must enter the domain name at the login page of IBM Cloud Orchestrator. To login to Business Process Manager, you must specify the domain name as a prefix of the username wherein the delimiter between domain name and username is a '/'. For example, a user ‘user1’ of domain ‘domain1’ must specify ‘domain1/user’. If you are a user who is within the default domain, you must authenticate only with your username. Users and projects are shown in Business Process Manager as users and groups. Any user or project of custom domains are prefixed with the domain name delimited by '/', that is, project ‘p1’ of domain ‘domain1’ appears as a group ‘domain1/p1’ in Business Process Manager. To ensure backward compatibility, users and projects of the default domain appear with its user and project name. As the user and the project name are prefixed by their
domain name in Business Process Manager with a '/' as delimiter, the user name, project name, and domain name must not contain a '/' character.

Note: In the current release of IBM Cloud Orchestrator user, project and domain names are case insensitive (in other words, if a user is created with the name 'George Windsor', then they would be able to login successfully as 'george windsor'). However, this is likely to change in a future release, so this fact should not be relied on.

If you are using an LDAP server to authenticate users, you can configure LDAP authentication to allow any corporate directory details to be specified on a domain-by-domain basis. For more information, see "Configuring LDAP authentication" on page 129.

You can work with your users, projects (only to manage users), and domains with the IBM Cloud Orchestrator user interface.

Model and terminology

The multitenancy model is based on the OpenStack identity service version 3. This is implemented by Keystone. This includes the following entities and relationships.

Domain

A domain is the highest entity in the identity model and represents a tenant as it is a container and namespace for projects and users of a customer. IBM Cloud Orchestrator allows segregation on both the domain level and the project level. Whether the domain concept is leveraged depends if the tenant should be allowed to organize itself and requires the role of a domain administrator. If the domain is a self-organized unit, create a domain and its domain administrator. A domain can have multiple projects and users. The project and users are owned by a domain. The domain administrator can manage the project and users and assign resources to them. If the customer is not a self-organized unit and the administrator of the service provider configures all projects, users and resources, the domain concept can be ignored and the Default domain can be used. The Default domain always exists.

User

A user represents the account of a person. You can login to IBM Cloud Orchestrator with a user account. A user account contains:

- a username
- password
- and email address

A user is unique within a domain. You can have two different users with the same name in two different domains. A user must always be member of at least one project and have a default project defined.

Project

A project is a container which owns resources. Resources can be:

- virtual machines
- stacks
The project is unique within a domain. This means you can have two projects with the same name in two different domains. A project can also have one or more users as members. With a membership you can access all resources owned by the project, so if you are a member of a project, you can access all resources owned by that project.

**Role**

A role grants you access to a set of management actions. IBM Cloud Orchestrator supports four different roles:

- **admin** – a super administrator in the cloud who has access across all domains
- **domain admin** – an administrator who has access and can manage resources, projects and users in the scope of it's domain.
- **catalog editor** – a user who can modify the catalog content and managing patterns in the scope of it's project.
- **member** – an end user who can perform any kind of self-service operation in the scope of it's project.

**Scope**

You can be member of one or multiple projects. As a user you will always work in the scope of a project. When you login, you will work on-behalf-of it's default project. If you are a member of multiple projects, you can switch across it's projects in the self-service banner.

**LDAP**

IBM Cloud Orchestrator can be configured to authenticate users with an LDAP or Active Directory. It is allowed to configure one LDAP for all domains or a specific LDAP per domain. If you log in to a domain with a LDAP configured you are authenticated against the LDAP of that domain.

**User roles in IBM Cloud Orchestrator**

Protect your cloud environment by using roles to control how different users interact with IBM Cloud Orchestrator. When you assign roles to users, you designate the types of objects that the users can access, and the tasks that the users can perform.

In IBM Cloud Orchestrator, the users and the projects are defined in the related OpenStack environment. When you create a user, you must assign a role to the user. The role is related to the project to which the user belongs. A user can have one or many roles within a project, and can have different roles in different projects.

The authority to access a type of object might not equate with the authority to access all instances of that object. In some cases, users can access an object instance only if they have been granted authority by the creator of that instance. See "Object-level access control" on page 208 for more information.

In IBM Cloud Orchestrator, you can use the following roles:
admin role
A user with this role can do the following tasks in the IBM Cloud Orchestrator environment:

- Create domains, projects, users, and roles.
- Assign other user roles. Configure product settings. Modify resources such as images, patterns, shared services, environment profiles, other catalog content, and deployed cloud content.
- Grant users access to projects. Grant projects access to regions and availability zones. Assign quotas to domains.
- Create environment profiles to group related cloud topology settings for easy deployment of virtual system patterns. Environment profile creators can also edit or delete any profile that they create, or to which they have access.
- Download audit data. Change auditing settings that are editable, such as the option to delete audit data from IBM Cloud Orchestrator after download.
- Do all of the tasks that a user with the domain_admin role can do.

Important: A user with the admin role has full privileges for all cloud resources, including all projects and all domains. Assign this role only to the cloud administrators, and only to users within the Default domain and admin project.

domain_admin role
A user with this role can do the following tasks in the IBM Cloud Orchestrator environment:

- View the details of the domain.
- View the projects, users, groups, offerings, and actions of the domain.
- Create, edit, and delete projects, users, groups, offerings, and actions that are associated with the domain.
- Manage the quota, availability zones, and networks for projects in the domain.
- Do all of the tasks that a user with the catalogeditor role can do.

catalogeditor role
A user with this role can do the following tasks in the IBM Cloud Orchestrator environment:

- Create virtual system patterns and virtual application patterns. Edit or delete any patterns that they create or to which they have access.
- Add objects to the IBM Cloud Orchestrator catalog. Modify or delete any catalog content that they create or to which they have access.
- Import image files.
- Create self-service offerings, self-service categories, and orchestration actions. Modify or delete any self-service offerings, self-service categories, and orchestration actions that they create or to which they have access.
- Do all of the tasks that a user with the member role can do.

member role
A user with this role can do the following tasks in the IBM Cloud Orchestrator environment:

- View and manage the virtual system instances, patterns, and catalog content to which they are granted access.
• Deploy virtual system patterns and virtual application patterns.

**Important:** A user with the *member* role cannot add, remove, or modify any items.

**Note:** The following roles, which are showed in the Administration user interface, are used only for the OpenStack services and not for IBM Cloud Orchestrator:

- member
- KeystoneAdmin
- KeystoneServiceAdmin
- sysadmin
- netadmin

### Object-level access control

IBM Cloud Orchestrator implements an object-level access control framework for some object types. Users must have specific levels of access permissions to view, or perform tasks with, instances of those objects. For example, a user with *Member* role can only deploy a particular pattern if the user belongs to a project that has been granted access to that pattern. Even a user with the *catalogeditor* role cannot modify a particular pattern unless he or she created that pattern, or belongs to a project that has been granted access to that pattern by the creator. Object-level access applies to the following types of objects:

- Environment profiles
- Virtual images
- Patterns
- Script packages
- Add-ons
- Virtual system instances
- Shared services instances
- Orchestration actions

The following table depicts the object-level access definitions in IBM Cloud Orchestrator. The *all or owner* definition applies to both the creator of the object instance and the users with *admin* role (which has complete access to every instance of every object that is created in the product). Object creators and users with *admin* role can assign the object permissions *read*, *write* or *all* to other projects.

**Table 23. Object level access definitions**

<table>
<thead>
<tr>
<th>Access level definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>You can see the object listed in the user interface panels and are able to view the details for this object.</td>
</tr>
<tr>
<td>Write</td>
<td>You can see the object listed in the user interface panels and are able to view and modify the details for this object.</td>
</tr>
<tr>
<td>All or owner</td>
<td>You can see the object listed in the user interface panels and are able to view and modify the details for this object. You are able to delete this object.</td>
</tr>
</tbody>
</table>
Table 23. Object level access definitions (continued)

<table>
<thead>
<tr>
<th>Access level definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>If you are not assigned access to the object, you cannot see the object listed in the user interface panels. You cannot perform any action associated with the object.</td>
</tr>
</tbody>
</table>

**Related tasks:**
- "Modifying access control list of an action” on page 323
- "Modifying access control of an offering” on page 320

**Regions, availability zones and quota**

This section describes regions, availability zones, and quota and their relationship in IBM Cloud Orchestrator.

A region is defined by a set of OpenStack services operating on the same hypervisor. A region can manage a set of KVM hosts or a VMware vCenter environment. It is not possible to manage different types of hypervisors within the same region.

An availability zone is a subset of compute resources within a region. A region can have many availability zones, but an availability zone can belong to only one region. Availability zones are used as the target for deployment and as a user, you must select the availability zone to place a virtual machine or stack.

IBM Cloud Orchestrator allows access of availability zones to Domains and Projects to allow an End User to manage them.

It is possible to define a quota to limit the allocation of resources in the cloud. The following definitions describe the different types of quota:

**Domain Quota**
The sum of all project quota must not exceed the domain quota. The domain quota is only enforced if the Domain Administrator creates a new project or modifies the quota of an existing project. If the domain quota is exceeded, the Domain Administrator is not allowed to create the project or increase the quota. The quota is not enforced for a Cloud Administrator.

**Project Quota**
The project quota is defined per region. In other words, the project can have different quota in different regions. Both a Cloud Administrator and a Domain Administrator can change the quota of a project in a region. The quota the Domain Administrator cannot exceed the overall domain quota while it is changing.

**Default Quotas**
Default quotas can be configured which are set for each new project. The domain default quota is a multiplication of a factor of the default project quota. Both default settings apply across domain and can only be configured by the cloud administrator.

**Environment Profile Quota**
The environment quota is used in IBM Workload Deployer to grant a
project to an availability zone. An environment profile can have a quota. It is marked obsolete because it is the same as a project quota. So, it is suggested to manage the project quota rather than the environment profile quota.

If a user requests a service that allocates resources in the cloud, only the quota its current project is checked and enforced. If the request exceeds the quota in that region, the request fails with an error message. It is ensured by definition, that the domain quota is not exceeded, as the Domain Administrator cannot give more quota to its projects. Therefore, only check the project quota during deployment.

**Network isolation**

The segregation of tenants also requires the isolation of tenant networks.

IBM Cloud Orchestrator supports two technologies to manage the networks within the cloud:

- OpenStack Nova network service
- OpenStack Neutron network service

Details of the services can be found in Chapter 2, “Installing,” on page 13 or in the OpenStack official documentation.

From a multi-tenancy perspective, the Neutron service provides the best capabilities, which allow tenants to manage their network topology in separate namespaces. This includes, overlapping IP scenarios, where two tenants can have two separate networks each having the same overlapping IP sub-range. The isolation and separation is handled the Neutron service.

The Cloud Administrator is responsible to manage the network topology. It is the service providers and Cloud Administrators responsibility to provide the external connectivity to the tenants.

In IBM Cloud Orchestrator, network isolation happens on the project layer, not on the domain layer. A network is typically owned by a project, and a project can have multiple networks.

The Domain Administrator can create private networks for projects within its domain.

The End Users, who request virtual machines on behalf of the current project can add one or more network interfaces. This ensures connectivity to one or more networks of the project.

This means that a tenant can manage their internal connectivity and can define a multi-tier application environment with different networks for database and application traffic.

However, the external connectivity must be ensured by the Cloud Administrator.

For more information about managing networks, see “Managing Nova networks” on page 107 and “Managing Neutron networks” on page 110.
Administering as domain administrator

Administer as a domain administrator.

A typical scenario for a Domain Administrator in a multi-tenancy environment is the onboarding of users into the domain. The administrator assigns the users to projects and ensures that they can deploy virtual machines and use cloud resources.

Managing domains

Log in to the Self-service user interface as a Domain Administrator.

Click CONFIGURATION > Domain and select Domains from the menu that is displayed, to see a list of domains that you can administer. To see more details about a particular domain, click the domain name. You can search for a particular instance by specifying the instance name or description in the search field. The instance table can be sorted using any column that has the sort icon.

Domain Administrators can manage projects, groups, users, actions, offerings, and categories. Domain Administrators can distribute the domain quotas among the projects in their domain.

Note: Domain Administrators cannot create domains, and they cannot change the domain quotas.

The Domain Administrator must perform the following steps to set up projects in the domain and to grant access to cloud resources to the domain users:

1. Create a project resource.
2. Ensure that the project has access to at least a deployment availability zone. This allows users in that project to access virtual images and deploy virtual servers when working within this project in the logged in domain. The availability zones that can be assigned to the project are the ones that have been previously assigned to the domain to which the project belongs to.
3. Set the quota in the project.
4. Create and add users to the project.

These steps are detailed in "Managing projects" and "Managing users" on page 217.

Managing projects

As a domain administrator you can manage the level of access for each project to IBM Cloud Orchestrator with the user interface.

You can search for a particular instance by specifying the instance name or description in the search field. The instance table can be sorted using any column that has the sort icon.
Creating a project:

As a domain administrator you can create a new project in a domain.

**Procedure**
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click **CONFIGURATION > Domain**.
3. Click **Domains** in the menu below the navigation menu.
4. Select the check box next to the domain you want displayed in the list.
5. Click **Create Project** in the **Actions** menu. The **Create Project** window is displayed.
6. Specify the name for the project.
7. Enter a description for the project.
8. Optional: By clearing the enabled check box, you disable and cannot authorize the domain. Selecting the enabled check box keeps the domain enabled so that you can authorize the project.
9. Click **OK**.

**Results**

A new project is created.

Enabling a project:

As a domain administrator you can enable a project in a domain.

**Procedure**
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click **CONFIGURATION > Domain**.
3. Click **Projects** in the menu below the navigation menu.
4. Select the check box next to the project you want displayed in the list.
5. Click **Enable Project** in the **Actions** menu. This option will only appear if the project is disabled.
6. Click **Confirm**

**Results**

A window appears at the top right of the screen confirming that the project has been enabled.

Edit a project:

As a domain administrator you can edit a project in a domain.

**Procedure**
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click **CONFIGURATION > Domain**.
3. Click **Projects** in the menu below the navigation menu.
4. Select the check box next to the project you want displayed in the list.
5. Click **Edit Project** in the **Actions** menu.
6. Specify the name of the project.
7. Optional: Enter a description and domain for the project. By clearing the enabled check box, you disable and cannot authorize the project. Selecting the enabled check box keeps the project enabled so that you can authorize the project.
8. Click OK.

Results
A window appears in the top right of the screen confirming that the project has been edited.

Disabling a project:
As a domain administrator you can disable a project in a domain.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Projects in the menu below the navigation menu.
4. Select the check box next to the project you want displayed in the list.
5. Click Disable Project in the Actions menu. A window is displayed asking if you want to confirm launching the action: Disable Project.
6. Click Confirm.

Results
A window appears in the top right confirming that the project has been disabled.

Deleting a project:
As a domain administrator you can delete a project in a domain.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Projects in the menu below the navigation menu.
4. Select the check box next to the project you want deleted.
5. Click Delete Project in the Actions menu.
6. A window appears asking if you want to delete the project. Click Confirm.

Note: Deleting the default project of a domain results in the domain quotas becoming empty because the domain quotas are a multiplier of the default project quotas. Refer to Setting the default domain quotas for more details on the domain quotas.

Results
A window appears at the top right of the screen confirming that the project has been deleted.
Creating a network for a project:

As a domain administrator you can create a network for a project in a domain.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Projects in the menu below the navigation menu.
4. Select the check box next to the project you want displayed in the list.
5. Click Create Network in the Actions menu.

   Note: The Create Network action in the Self-service user interface is limited to Nova networks only. To create a Neutron network, use the Administration user interface.
6. Select the region from the drop down menu
7. Use the following network parameters:
   - Name: the name of the network
   - Mask (CIDR): the definition of the IP range of the network, e.g. 10.10.10.x/24 for IP addresses between 10.10.10.1 and 10.10.10.255.
   - VLAN/VXLAN ID: the VLAN ID that should be used to separate the network traffic from other networks, e.g. 1001.
   - Bridge interface: the interface on the cloud management servers that should be used for the bridge, e.g. eth0. Only required in regions having nova-network configured. Not required for regions having neutron configured.
   - Gateway: the gateway of the subnet, e.g. 10.10.10.1. By default the first IP in the range is used.
   - DNS: the primary and secondary DNS servers to be used.
8. Click OK.

Results
The network is created for and is owned by the selected project. Only members of the project can use the network.

Deleting a network from a project:

As a domain administrator you can delete one or multiple networks from a project.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Projects in the menu below the navigation menu.
4. Select the check box next to the project you want displayed in the list.
5. Click Delete Network in the Actions menu.
6. Select the network you want to delete.
7. Click OK.

Results
The network is deleted from the region it has been created.
Modify the availability zones of a project:

As a domain administrator you can grant and revoke access of availability zones to a single project in a domain.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Projects in the menu below the navigation menu.
4. Select the check box next to the project you want displayed in the list.
5. Click Modify Availability Zones in the Actions menu. The Availability Zones of Domain and the Availability Zones of Project are listed in the following format: Availability_Zone – Region.
6. Complete one or more of the following options to modify the user availability zones of a project:
   • To assign a zone to a domain from the list of Availability Zones of Domain, select an availability zone by selecting the check box beside it, then click the >> button. The selected zone moves to the Availability Zones of Project list.
   • To return an Availability Zone of Project to an Availability Zone of Domain, select an availability zone by selecting the check box beside it, then click the >> button beside the zone name.
7. Click OK.

Results

The changes you made to the availability zones of the project has been saved.

Modify the quota of a project:

As a domain administrator you can modify the quota of a single project in a domain.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domains.
3. Click Projects in the menu below the navigation menu.
4. Select the check box next to the project you want displayed in the list.
5. Click Modify Quota in the Actions menu.
6. Select the region from the drop down menu.
7. Click Next.
8. The quota dialog box contains values for the number of cores, the number of instances, amount of memory and the number of floating Ips. Enter a value for each.

   Note: The sum of all project quota of a domain can not exceed the overall quota of the domain. The validate button checks if that condition is met. If the condition is not met, the quota can not be changed. To view the overall domain quota and the quota remaining in the domain, click Show Domain Quota.
9. Click OK.
Results

The changes you made to the quota of the project has been saved.

Modifying users in a project:

As a domain administrator you can add and remove users of a single project in a domain.

Procedure

1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Projects in the menu below the navigation menu.
4. Select the check box for the project that you want to edit.
5. In the Actions menu, click Modify Users.

Note: The Modify Users page shows the following lists of users:

- Users in Domain: Users in the current domain who are not assigned to the selected project.
- Users in Project: Users assigned to the current project, with roles assigned. The user roles are also shown.
6. To assign a user to a project, select the check box beside the user, then click >>. The selected user moves to the User in Project list.
7. To remove a user from a project, select the check box beside the user, then click <<. The selected user moves to the Users in Domain list.
8. To edit the role assignment of a user in the User in Project list, click the Role column for the user. Select one or multiple roles from the roles list.
9. Click OK.

Results

The changes you made to user roles and user assignments for a project have been saved.

Managing groups

As a domain administrator you can manage groups in IBM Cloud Orchestrator with the user interface.

Creating a group:

As a domain administrator you can create a new group in a domain.

Procedure

1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Domains in the menu below the navigation menu.
4. Select the check box next to the domain you want displayed in the list.
5. Click Create Group in the Actions menu. The Define Group window is displayed.
6. Specify the name for the group.
7. Optional: Enter a description and domain for the group.
8. Click OK.

Results
A new group is created.

Deleting a group:
As a domain administrator you can delete one or multiple groups in a domain.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Groups in the menu below the navigation menu.
4. Select the check box next to the group you want displayed in the list.
5. Click Delete Group in the Actions menu.
6. Click Confirm in the window that opens.

Results
A window appears in the top right of the screen confirming that the group has been deleted.

Managing users
As a domain administrator you can manage the level of access for each individual user to IBM Cloud Orchestrator with the user interface.

You can search for a particular instance by specifying the instance name or description in the search field. The instance table can be sorted using any column that has the sort icon.

Creating a user:
As a domain administrator you can create a new user in a domain.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Domains in the menu below the navigation menu.
4. Select the check box next to the domain you want displayed in the list.
5. Click Create User in the Actions menu. The Create User window is displayed.
6. Specify the name for the user, the default project to assign the user to and the users role in that project.
7. Optional: Enter an email, password and domain for the user. By clearing the enabled check box, you disable and cannot authorize the user. Selecting the enabled check box keeps the user enabled so that you can authorize the project.
8. Click OK.

Results
A new user is created and appears in the User view. This action applies only to a single domain.
Deleting a User:

As a domain administrator you can delete one or multiple users in a domain.

Procedure
1. Log in to the Self-service user interface as a Domain Administrator.
2. In the navigation menu, click CONFIGURATION > Domain.
3. Click Users in the menu below the navigation menu.
4. Select the check box next to the user you want displayed in the list.
5. Click Delete User in the Actions menu.
6. Click Confirm in the window that opens

Results
A window appears in the top right of the screen confirming that the user has been deleted.

Administering as cloud administrator

Administer as a cloud administrator.

A common scenario for a Cloud Administrator is the onboarding of a customer or organization, which must have the required administrator roles and cloud resources assigned to be up and running. For details, see “Managing a domain.”

Note: The administration user interface does not support filtering with some special characters for example: $ ^ ( ) . [ \ ? ] %

Managing a domain

You can manage domains in IBM Cloud Orchestrator with the Administration user interface.

About this task

Domains represent a customer or an organization in a multi-tenant environment. Perform the following steps for onboarding a customer in this environment. You must be assigned the admin role to perform this procedure.

Procedure
1. Create a domain resource. This step automatically creates a default project for the domain to facilitate user onboarding.
2. Ensure that the domain has access to at least one deployment availability zone. This allows users in that domain to access virtual images and deploy virtual servers when logged into the domain projects. The availability zones assigned to the domain will then be visible to be assigned to projects within the domain.
3. To delegate the domain administration, ensure that at least one user is assigned to the domain with domain_admin role. With this role, the Cloud Administrator can delegate the administrative tasks of the domain to the Domain Administrator who can then start creating projects and assigning users.

Note: The default OpenStack Cloud Administration domain Default should not be disabled. If it is, you are unable to log in to the Orchestrator and Administration UI as default Cloud Administrator. It makes the default Cloud
Administrator domain and projects not valid. If you disabled the domain, you can enable it again in one of the following ways:

- Send an HTTP request as follows:
  ```
curl -i -X PATCH http://<HOST>:35357/v3/domains/default
-H "User-Agent: python-keystoneclient"
-H "Content-Type: application/json" -H "X-Auth-Token:<TOKEN>"
-d '{"domain": {"enabled": true, "id": "default", "name": "Default"}}'
```
- Update the domain to be enabled with the python client using V3 keystone.

Creating a domain:

The Cloud Administrator creates domains to organize projects, groups, and users. Domain administrators can update and delete resources in a domain.

**Procedure**

1. Log in to the Administration user interface as the Cloud Administrator.
2. In the left navigation pane, click **Admin > Identity Panel > Domains**. The Domains page opens.
3. Select **Create Domain**. The Create Domain window is displayed.
4. Specify the domain name and, optionally, the domain description.
5. Optional: Clear the **Enabled** check box to disable the domain. If the domain is disabled, the Domain Administrator cannot create, update, or delete resources related to the domain. New domains are enabled by default.
6. Click **Create Domain**.

**Results**

A message is displayed, indicating that the domain is created successfully. A project called **Default** is automatically created for the new domain.

Assigning a zone to a domain:

Assigning a zone to a domain enables users within a zone to access a specific domain.

**Before you begin**

You must be logged in with the **admin** role to complete these steps.

**Procedure**

1. Log in to the Administration user interface as the Cloud Administrator.
2. Open the domains page by clicking **Admin > Identity Panel > Domains** in the navigation pane.
3. In the domains page, find the entry for the domain and click **More > Edit** in the **Actions** column to open the Edit Domain window.
4. Click the **Availability Zones** tab. The **Available Zones** and the **Assigned Zones** are listed in the following format: **Zone_Name - Region_Name**
5. To assign a zone to a domain, from the list of **Available Zones**, click the plus button beside the zone name. The selected zone moves to the **Assigned Zones** list. To return an **Assigned Zone** to an **Available Zone**, select the minus button beside the zone name. Use the **Filter** field to search for specific zones.
6. When you have assigned all zones, click **Save**.
Results

A message indicates that the domain is modified successfully.

Setting the default domain quotas:

The Cloud Administrator sets the default domain quota for certain resources to specify the maximum amount of the resource that is available to the domain. The Domain Administrator can then distribute that quantity among all the projects in the domain.

Before you begin

You must be assigned the admin role to complete these steps.

About this task

The Cloud Administrator can assign a default project quota to certain resources, as described in “Configuring project quotas” on page 225.

To generate the default domain quota values, IBM Cloud Orchestrator multiplies the corresponding default project quota value by the projects_per_domain variable.

Note: The projects_per_domain variable is a multiplier that IBM Cloud Orchestrator applies to the default project quotas, to calculate the default domain quotas. The projects_per_domain variable does not specify the maximum number of projects that can be created in a domain.

The default value of the projects_per_domain variable is 5. The Cloud Administrator can change the value of the projects_per_domain variable, which consequently changes the default domain quotas, as follows:

Procedure

1. Log in to the Administration user interface as the Cloud Administrator.
2. Log in to Central Server 2 as a root user.
3. Add the following lines to the /etc/openstack-dashboard/local_settings file:
   ```python
   SCO_CONFIG = {
               'projects_per_domain': <number of projects>,
   }
   ```
4. Restart the HTTPD service:
   ```bash
   service httpd restart
   ```

Editing the domain quotas:

The Cloud Administrator can change the quotas of a domain to set limits on the operational resources that a Domain Administrator can distribute among all the projects in the domain.

Before you begin

You must be assigned the admin role to complete these steps.
Procedure

1. Log in to the Administration user interface as the Cloud Administrator.
2. In the navigation pane, click Admin > Identity Panel > Domains.
3. On the Domains page, find the entry for the domain that you want to modify. In the Actions column for that entry, click More > Edit.
4. In the Edit Domain window, click the Quota tab.
5. Edit the quota values as required.
6. Click Save.

Results

A message is displayed, indicating that the quotas were saved to the domain successfully.

Modifying the list of domain administrators:

You can add or remove users from the list of Domain Administrators to control a domain.

About this task

To modify the list of domain administrators who are assigned to a domain, complete the following steps:

Procedure

1. Log in to the Administration user interface as the Cloud Administrator.
2. In the navigation pane, click Admin > Identity Panel > Domains.
3. In the Domains page, select the entry for the domain. In the Actions column, click More > Edit. The Edit Domain window opens.
4. Click the Domain Administrators tab.

Note: The Domain Administrators tab shows the following lists of users:
- All Available: Users that are members of the domain but are not Domain Users
- Domain Administrators: Users who are Domain Administrators for the selected domain.
5. To add a Domain Administrator, click +. The user is promoted from Domain User to Domain Administrator for the default project only. You must manually add the Domain Administrator user to all other projects in the domain, as described in “Modifying user assignments for a project” on page 227.
6. To remove a Domain Administrator, click -. The user is demoted from Domain Administrator to Domain User for all projects in the domain, but is not removed from any projects.
7. Click Save.

Results

The changes you made to the list of domain administrators has been saved.
Setting a domain context:

The domain context is for Cloud administrators to refine the context that they are accessing. Cloud administrators can limit the scope to one domain, rather than having visibility across all domains. This allows the Cloud administrator to identify the projects, users, groups and roles that are associated with a domain.

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. In the left navigation pane, click Admin > Identity Panel > Domains.
3. In the domains page, find the entry for the domain and click Set Domain Context

Results
The domain page title changes to <domainName>:Domains. Selecting the Projects, Users, Groups or Roles web pages will only display details for the selected domain context.

Clearing the domain context:

Cloud administrators can clear the scope of all domains, enabling visibility across all domains.

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. In the left navigation pane, click Admin > Identity Panel > Domains.
3. In the domains page, select Clear Domain Context from the top right-hand corner.

Results
All domains are visible.

Managing projects

You can manage the level of access for each project to IBM Cloud Orchestrator with the user interface.

Before you begin

You must be assigned the admin role to perform these steps.

Creating a project:

You can assign individual zones to each domain in IBM Cloud Orchestrator with the administrator user interface.

Before you begin

Set the domain context via Setting the domain context

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. Open the projects page by clicking Admin > Identity Panel > Projects in the navigation pane.

3. Click Create Project. The Create Project window is displayed.

4. Specify the name for the project.

5. Optional: Enter a description for the project.

6. Optional: By clearing the Enabled check box, you disable and cannot authorize the domain. Selecting the Enabled check box keeps the domain enabled so that you can authorize the domain.

7. Click Create Project.

Results

A message indicates that the project is created successfully.

Enabling a project:

Enabling a project allows you to set that project as your default project. the action only appears if the project is disabled.

Procedure

1. Log in to the Administration user interface as the Cloud Administrator.

2. Open the projects page by clicking Admin > Identity Panel > Projects in the navigation pane.

3. In the projects page, find the entry for the project and click More > Edit Project in the Actions column.

4. In the Edit Project window, click the Enabled check box so that the box contains a tick symbol.

What to do next

A message is displayed indicating that the project is enabled.

Editing a project:

You can modify the name and description of a project.

Procedure

1. Log in to the Administration user interface as the Cloud Administrator.

2. Open the projects page by clicking Admin > Identity Panel > Projects in the navigation pane.

3. In the projects page, find the entry for the project and click More > Edit Project in the Actions column.

4. In the Project Info tab, edit the name and description of the project.

Results

A message is displayed indicating that the project information has been modified.
Disabling a project:

Disabling a project in a domain means that users who previously had that project set as their default cannot log in to it anymore. Other users also cannot switch to this project anymore.

Procedure

1. Log in to the Administration user interface as the Cloud Administrator.
2. Open the projects page by clicking Admin > Identity Panel > Projects in the navigation pane.
3. In the projects page, find the entry for the project and click More > Edit Project in the Actions column.
4. In the Edit Project window, de-select the Enabled check box so that the box is empty.

Results

A message is displayed indicating that the project is disabled.

Deleting a project:

Delete a project in the Administration user interface as the Cloud Administrator

Procedure

1. Log in to the Administration user interface as the Cloud Administrator.
2. Open the projects page by clicking Admin > Identity Panel > Projects in the navigation pane.
3. Find the entry for the project that you want to delete. In the Actions column for that entry, click More > Delete Project.

Note: Deleting the default project of a domain results in the domain quotas becoming empty because the domain quotas are a multiplier of the default project quotas. Refer to Setting the default domain quotas for more details on the domain quotas.

Results

A message is displayed, indicating that the project has been deleted.

Assigning a zone to a project:

Assigning a zone to a project enables users within a zone to access a specific project.

Procedure

1. Log in to the Administration user interface as the Cloud Administrator.
2. Open the domains page by clicking Admin > Identity Panel > Domains in the navigation pane.
3. In the domains page, find the entry for the domain and select Set Domain Context in the Actions column. The Identity Panel group is now in the context of the selected domain and the Domains page is also changed. You are now working within the context of the domain that you created.
5. In the **Actions** column in the table for the project, click **More > Edit Project**.

6. Click the **Availability Zones** tab. The available zones and the assigned zones are listed in the following format: **Zone_Name - Region_Name**.

7. To assign a zone to a domain, from the list of **Available Zones**, click the plus button beside the zone name. The selected zone moves to the **Assigned Zones** list. To return an **Assigned Zone** to an **Available Zone**, select the minus button beside the zone name. Use the **Filter** field to search for specific zones.

8. When you have assigned all zones, click **Save**.

**Results**

A message indicates that the project is modified successfully.

**Configuring project quotas:**

The Cloud Administrator can configure the project quotas in OpenStack.

**About this task**

Log in to the Administration user interface as the Cloud Administrator. Use the command-line interface to set the default project quotas and to change the project quotas in OpenStack. You can specify project quotas for several different resources, such as CPUs and memory. For more information, see the [OpenStack documentation](http://docs.openstack.org/user-guide-admin/content/cli_set_quotas.html).

**Reassigning VM instances to a project:**

Reassigning VM instances to a project enables VMs which have been loaded to a default project to be assigned to the project of a user who owns them.

**Before you begin**

In order to reassign instances, you must have an admin role on the source project containing the instances to be reassigned.

**Procedure**

1. Log in to the Administration user interface as a Cloud Administrator.
2. In the navigation pane, click **Project > Instances**.
3. Find the instances to be reassigned and select the check boxes beside their names.
4. Click **Reassign Instances**.

   **Note:** **Reassign Instances** is only visible for a VMware region.

5. Selected instances contain a list of the instances selected from the Instances table. The following options are available:
   - To deselect an instance, click on the instance in the list box. At least one instance must be selected.
   - To select all instances press **Ctrl-Shift-End**.

6. In the Reassign Instances window, select the **Target Domain** where the instances are to be assigned.

7. Select **Target Project** where the instances are to be assigned.

8. Click **Reassign**.
Results

A message is displayed indicating that the instances have been reassigned successfully from the source domain and project to the target domain and project.

Note: When a VM instance is reassigned from one project to another, the resources that are associated with the VM (such as networks, IPs, flavors) are owned by the source project. If there are access issues to these resources from the new project, you need to recreate the resources on the new project.

Customizing and extending the user interface:

Role based directory structure:

Dashboards are role based. This allows service designers to have their dashboard extension content available to specific roles.

The directory structure that the extension content is deployed into is based on the SCO roles in the following table. The four out of the box SCO roles which the extension is compatible with are:

<table>
<thead>
<tr>
<th>Name of role directory</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>member</td>
<td>End User</td>
</tr>
<tr>
<td>admin</td>
<td>Cloud Administrator</td>
</tr>
<tr>
<td>domain_admin</td>
<td>Domain Administrator</td>
</tr>
<tr>
<td>catalogeditor</td>
<td>Service Designer</td>
</tr>
</tbody>
</table>

The dashboard folder is the parent extension directory. It contains a folder for each of the four roles. It is located on the icoui server at the following location: `{server_location}/etc/dashboard`

e.g. `{server_location}/etc/dashboard/admin`

When a service designer wants to make a new dashboard available to an admin user, for example, they add the extension html file to the: `{server_location}/etc/dashboard/admin` directory

Navigation elements and extension naming conventions:

Naming extension files is important as these names drive the names of the navigation elements where the dashboards are accessed from.

Navigation elements should be meaningful so that you can easily locate important content in the UI. As these dashboard extension navigation elements are driven from the names of the dashboard extension files, these file names should be meaningful. Dashboard extensions appear as submenu items under the DASHBOARD menu and take the name of the extension file as their menu label. The label is the complete file name with any ordinal position specified and the file extension removed.

The following table describes an example of file names and their respective menu labels:
Table 25.

<table>
<thead>
<tr>
<th>File</th>
<th>Menu label</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 - Admin Extension Example.html</td>
<td>Admin Extension Example</td>
</tr>
<tr>
<td>02 - Member Extension Example.html</td>
<td>Member Extension Example</td>
</tr>
<tr>
<td>Network Dashboard.html</td>
<td>Network Dashboard</td>
</tr>
<tr>
<td>Performance Dashboard.html</td>
<td>Performance Dashboard</td>
</tr>
</tbody>
</table>

Ordinal numbering of extension files is included so that you can control the order in which the extension labels appear in the DASHBOARD menu. The ordinal numbering format convention is a number, then a hyphen (-), then the file name. Any file name starting with this pattern is placed in this relative numbered position in the sub menu. The pattern is stripped from the file name when constructing the label. If an ordinal numbering format convention is not used when naming extension files the files is added alphabetically.

Packaging and deployment:

A dashboard extension made available via the market place must be packaged as a compressed zip or tar.gz file.

The package structure must match the role based directory structure mentioned in the previous section. For example:

```
extensionExample.zip
  + dashboard
    + admin
      + 01 - My Admin Extension Example.html
```

To deploy the extension, extract the compressed file to `{server_location}/etc`. To have the new extension content appear in the UI, restart the icoui server. Alternatively, enter the following link in your browser to clear the navigation cache: `<scuiHostName:port>/dashboardextension/clearcache`

**Note:** Before deploying a dashboard extension ensure that there are no extension files that have already been deployed previously in the extension directories with the same name. Any deployed file with the same name as one file contained in the package will get overwritten otherwise.

Modifying user assignments for a project:

You can assign users to additional projects or update and remove assignments. You can also specify the roles that the user will have for the project.

**About this task**

To modify user assignments for a project, complete the following steps:

**Procedure**

1. Log in to the Administration user interface as the Cloud Administrator.
2. In the navigation pane, click **Admin > Identity Panel > Projects**.
3. Click **Modify Users** for the project that you want to modify.

**Note:** The Edit Project window shows the following lists of users:

- **All Users**: Users who are members of the domain but are not members of this project.
• **Project Members**: Users who are members of this project and associated roles. This list also shows the roles that are assigned to each project member.

4. To assign a user to this project, click +. The user is moved from the **All Users** list to the **Project Members** list.

5. To remove a user from this project, click -. The user is moved from the **Project Members** list to the **All Users** list.

6. To change the roles that are assigned to a project member: in the **Project Members** list, expand the role list for the user, and select the roles.

7. Click **Save**.

**Results**

The changes you made to user assignments for a project has been saved.

**Managing groups**

You can manage the level of access for each group to IBM Cloud Orchestrator with the user interface.

**Deleting a group:**

Delete one or more groups in a domain.

**Procedure**

1. Log in to the Administration user interface as the Cloud Administrator.

2. Open the groups page by clicking **Admin > Identity Panel > Groups** in the navigation pane.

3. Find the entry for the project that you want to delete. In the Actions column for that entry, click **More > Delete Group**.

**Results**

A message is displayed, indicating that the group has been deleted.

**Managing users**

You can manage the level of access for each individual user to IBM Cloud Orchestrator with the user interface.

**About this task**

**Note:**

• When the current user password is modified, after re-login, the user is redirected to the **Edit User** dialog, where the user has to click **Cancel**.

• You cannot remove the email address by editing a user in the Administration user interface. To remove the email address, use the following command:

```bash
keystone user-update --email "" <USER_NAME or USER_ID>
```
Creating a user:

You can manage the level of access for each user in IBM Cloud Orchestrator. Users can be assigned to different roles on different projects.

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. In the navigation pane, click **Admin > Identity Panel > Users**.
3. Click **Create User**. The Create User window is displayed.
4. Specify the required parameters, and then click **Create User**.

Results

A message indicates that the user is created successfully.

Deleting a user:

You can delete one or multiple users in a domain.

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. Open the users page by clicking **Admin > Identity Panel > Users** in the navigation pane.
3. Find the entry for the user that you want to delete. In the Actions column for that entry, click **More > Delete User**.

Results

A message is displayed, indicating that the user has been deleted.

Managing volumes

You can attach volumes to instances to enable persistent storage.

Procedure
1. Log in to the Administration user interface as a Cloud Administrator.
2. Click **CONFIGURATION > System Panel > Volumes**.
3. Create and manage volumes as described in the [Create and manage volumes topic](#) in the OpenStack documentation.

Managing networks

As a domain administrator you can manage networks in IBM Cloud Orchestrator with the user interface.
Creating a network:

As a domain administrator, you can create a new network in a domain.

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. In the left panel, select PROJECT > Network > Networks.
3. Click Create Network.
4. Enter the following information:
   - Network Name
     Name for the new network.
     **Note:** for a VMWare region, the name of the new network must match
     the name of the network as defined in the vCenter Server that is being
     managed.
   - Create Subnet
     Select to create a subnet.
   - Subnet name
     Name for the subnet.
   - Network Address
     IP address for the subnet.
   - IP Version
     IP version for the network.
   - Gateway IP
     Optional. IP address for a specific gateway.
   - Disable Gateway
     Select to disable a gateway.
   - Enable DHCP
     Select to enable DHCP.
   - Allocation Pools
     IP address pool.
   - DNS Name Servers
     Name of DNS Servers.
   - Host Routes
     IP address of host routes.
5. Click Create.

Results
A message appears indicating that the network is created successfully.
Deleting a network:

As a domain administrator, you can delete one or multiple networks in a domain.

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. In the left panel, select PROJECT > Network > Networks.
3. Select the networks that you want to delete.
4. Click Delete Networks.

Results
A window appears in the top right of the screen confirming that the networks have been deleted.

Modifying a network:

As a domain administrator, you can edit one or multiple networks in a domain.

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. In the left panel, select ADMIN > Network > Networks.
3. Select the networks that you want to edit.
4. Click Edit Networks.

Results
A window appears in the top right of the screen confirming that the networks have been modified.

Adding a subnet to an existing network:

As a domain administrator, you can add a subnet to an existing network.

Procedure
1. Log in to the Administration user interface as the Cloud Administrator.
2. In the left panel, select ADMIN > Network > Networks.
3. Select the networks for which you want to add subnets.
4. Click More > Add Subnet.
5. Click Edit Networks.
6. Enter the subnet name, network address, Gateway IP and click Create.

Results
A window appears in the top right of the screen confirming that the networks have been modified.
Using the Public Cloud Gateway

The Public Cloud Gateway is installed on the Central Server 2 machine during the installation of IBM Cloud Orchestrator.

Public Cloud Gateway overview

The Public Cloud Gateway is a web application that provides an OpenStack API compatibility layer so the Amazon Elastic Compute Cloud (Amazon EC2), IBM SoftLayer, and non-IBM supplied OpenStack, work like the standard OpenStack Nova, Cinder, and Glance instances.

Note: Public Cloud Gateway provides a subset of the available OpenStack API. See “OpenStack API support” on page 234

The Public Cloud Gateway is automatically installed on the Central Server 2 machine as part of the IBM Cloud Orchestrator installation process. For more information about installing IBM Cloud Orchestrator, see the related installation guide. You can deploy and manage virtual machines and storage across both private and public clouds, for example, Amazon EC2.

Note: With the Public Cloud Gateway, higher level components such as Workload Deployer and the IBM Cloud Orchestrator User Interface operate normally with the public clouds with no changes required to support them.

Note: The Public Cloud Gateway can also be used to manage non-IBM supplied OpenStack. See the related section for more information.

See the related topic for more information about how the Public Cloud Gateway fits into the IBM Cloud Orchestrator product architecture.

Review the list of capabilities and limitations for the Public Cloud Gateway.

Related tasks:
Installing
Follow this procedure to install IBM Cloud Orchestrator.

Capabilities and limitations

Review the capabilities and limitations available with the Public Cloud Gateway.

New features and functions
- Project-level quota support
- Multi-tenancy support
- Deploy single virtual server offering
- Dashboard support
- Assigned Resources view support
- Offerings for key management: register and unregister key
- Attach additional volume

General limitations
- IBM Cloud Orchestrator Administration (Project and Admin section) is not supported for Public Cloud Gateway managed regions.
- Identity Panel must be used to configure Availability-Zones to Domains and Projects.
"Deploy a cloud Service using stacks" (OpenStack Heat deployment) is not supported for Public Cloud Gateway managed regions.

- Unable to import images from sources outside of public cloud repositories. If special images are required by patterns, this scenario is not supported.
- Supports only single predefined Dynamic Host Configuration Protocol (DHCP) network.
- Supports single or dual NIC depending on remote IaaS (Amazon EC and NIOS single NIC, IBM SoftLayer 1 or 2 NIC(s) depending on configuration).
- No Windows operating system support.
- SoftLayer only supports to resize the CPU and RAM and not the Primary Disk.
- SoftLayer does not support user data.

Public Cloud Gateway capabilities within IBM Cloud Orchestrator using Workload Deployer

- vSysClassic using cloud-init and scp-init images
- Deploy a virtual system.
- Stop and start a virtual system instance.
- Remove a virtual system instance.
- Script packages support is available.
- Configure administrator password on a virtual machine with images containing the Activation Engine.
- User add-on support is available.
- Disk add-on is available for virtual system parts

Public Cloud Gateway limitations within IBM Cloud Orchestrator using Workload Deployer

- Workload Deployer vSysNext is not supported.
- Network Interface Controller (NIC) add-on is not supported.
- Setting root password not supported for scp-init images.

Public Cloud Gateway capabilities within IBM Cloud Orchestrator using OpenStack

- Deploy a single virtual server offering.
- Stop and start a virtual system instance.
- Attach and detach a volume.
- Remove a virtual system instance.
- Execute script packages.
- Single predefined Dynamic Host Configuration Protocol (DHCP) network only.

Public Cloud Gateway limitations within IBM Cloud Orchestrator using OpenStack

- Unable to retrieve hypervisor resource information when used in a non-IBM supplied OpenStack configuration.
- OpenStack API support. The Public Cloud Gateway supports a limited set of OpenStack APIs.

Volume provider limitations

- For NIOS regions, no name or description is supported.
For Amazon EC2, name is supported and description is not supported.
For SoftLayer, name is supported and description is not supported. If no name is supplied, a name is created in the form HybridStorage-<volume_size>GB-<date>.

OpenStack API support:
The Public Cloud Gateway supports a limited set of OpenStack APIs.

Supported OpenStack APIs include:
• OpenStack Nova
• OpenStack Glance
• OpenStack Cinder

Supported OpenStack Nova API
• Create (boot) a virtual machine with:
  – SSH key
  – Availability zone
  – Single DHCP network
• List virtual machine instance:
  – Filter by status
  – Filter by SSH key name
• Delete virtual machine
• Start / stop virtual machine
• Show virtual machine detail
• List images
• Show image detail
• List Availability Zones
• List Networks providing a single DHCP network
• List Extensions
• List Flavors
• Show Flavor details
• Get version info
• Get Limits
• Show Network providing a single DHCP network
• Query quota for tenant
• Query quota defaults for tenant
• Set quota for tenant
• Delete quota of tenant
• Attach Volume
• Detach Volume
• Create key pair providing the SSH key import within the request
• List key pairs
• Delete key pair

Supported OpenStack Glance API
• List images
• Show image detail
Supported OpenStack Cinder API

- Create volume
- List volumes
- Filter by status
- Show volume detail
- Delete volume
- List volume types
- Single hard-coded entry

**Configuring the Public Cloud Gateway: Overview**

The Public Cloud Gateway is deployed as part of the IBM Cloud Orchestrator installation on the Central Server 2 machine. However, the Public Cloud Gateway is not enabled by default and certain updates to the configuration files are required before you use the Public Cloud Gateway.

**Before you begin**

Ensure that the prerequisites are satisfied.

**About this task**

To configure the Public Cloud Gateway, set up the following configuration files:

- admin.json
- config.json
- credentials.json
- flavors.json

By default, these files are located on Central Server 2 in the /opt/ibm/pcg/etc directory. When the Public Cloud Gateway is installed, default values for all parameters are provided except cloud access keys. Administrators must change only these settings in the configuration files that affect their particular setup.

The configuration of the Public Cloud Gateway is slightly different depending on the actual remote cloud:

- "Configuring Public Cloud Gateway for Amazon EC2" on page 252
- "Configuring the Public Cloud Gateway for SoftLayer" on page 257
- "Configure the Public Cloud Gateway to manage non-IBM supplied OpenStack" on page 262

There are a set of "Common configuration tasks" on page 240 that are common across all supported remote clouds.

**Related reference:**

- "Region names displayed incorrectly in the Virtual Image page" on page 267
- "Unable to connect to a public cloud due to missing credentials" on page 268
- "Loss of functionality in Public Cloud Gateway cloud groups" on page 267

A known issue exists where IBM Cloud Orchestrator removes the name after an " " in the region name when registering images.

In the Public Cloud Gateway, you might receive the error Unable to connect to public cloud due to missing credentials.

Loss of functionality may occur in Public Cloud Gateway cloud groups in IBM Cloud Orchestrator, where there has been heavy load on the Public Cloud Gateway cloud groups.
SSH key management

The Public Cloud Gateway provides the capabilities for SSH key management, that is, OpenStack key pairs.

For Amazon EC2 and SoftLayer, you can support register and deregister SSH key offerings in the catalog.

For information about SSH keys, see the description of the register and unregister key offerings:

- [“Registering a key pair” on page 315](#)
- [“Unregistering a key pair” on page 315](#)

Assumptions:

- Key pairs are subject to quota. A create key pair request might fail due to reaching the quota limits per project.
- Key pairs are subject to multi-tenancy. Key pairs are scoped on a per project basis. Known Limitation: IBM SoftLayer does not allow storing the same SSH key multiple times under different key pair names.

Multi-tenancy support

The Public Cloud Gateway provides capabilities for multi-tenancy.

These capabilities are in addition to the general multi-tenancy capabilities in the core product.

The Public Cloud Gateway contains the following capabilities related to multi-tenancy:

- Supports non-default domains and projects
- Limits the view of resources on project scope
- Creates resources in scope of a project
- Supports quotas on a per project and per region basis

The following concepts exist within the Public Cloud Gateway:

**Share an Amazon AWS EC2 account across projects using single set of credentials**

When a resource is created in Amazon AWS EC2, it is created within a project scope. This means that you can only see the resource that belongs to the project you are logged into. This feature is transparent in the Public Cloud Gateway.

**Note:** Resources are either tagged by project ID or separated by namespace. If you log on to the Amazon AWS EC2 management console, you can see the tags and namespaces for the various resources.

**Share a cloud account across projects using dedicated credentials per project**

The Public Cloud Gateway contains a configuration file that maps the credentials used for a region to a project. This feature allows you to supply different logon credentials on a per project and per region basis.

**Note:** The administrator of the IaaS account must ensure via ACL that the different user IDs have permissions to perform the actions and do not see the resources of the other user IDs.
Limitations
- The multi-tenancy support does not provide a physical segregation of resources because they still belong to the same account. It provides different views of the account on a per project basis.
- The network is shared across an account.
- Storage comes from a shared pool

Quota support overview
The Public Cloud Gateway provides capabilities for quota management.

The following are the types of quota definitions in public cloud gateway:
- A single global default quota used if no project level definitions are set.
- Per region and project quota set managed by the administration view in the project detail section.

The following quotas are supported in public cloud gateway. This is only a subset of the OpenStack defined set.

- **instances**
  Total number of instances that can be provisioned

- **cores**
  Total number of cores that can be consumed

- **ram**
  Total RAM size in MB that can be consumed. Must be larger than the "gigabytes" quota value

- **gigabytes**
  Largest size of a single volume in gigabytes

- **volumes**
  Number of volumes that can be created

- **key_pairs**
  Number of key pairs that can be created

The following actions can be done on quotas:
- Set global quota defaults. Maintained within the public cloud gateway configuration.
- Query and Set per project quota defaults. Maintained within the administration view in the project detail section.
- Delete project quota. Only Supported only via OpenStackAPI call. Not supported via the IBM Cloud Orchestrator UI.

Assumptions:
- Administrator of the Public Cloud Gateway region must ensure that the sum of the per project quota does not exceed the region capability.
- Quotas are enforced on a per project and region base.
Upgrading

There are some considerations for the Public Cloud Gateway when you upgrade.

The Public Cloud Gateway includes some new capability for multi-tenancy for Amazon EC2. See the following topics:

- "Multi-tenancy support" on page 236
- "Configuring Public Cloud Gateway for Amazon EC2" on page 252 to set up the credentials.json

The caching system of the Public Cloud Gateway has changed. See "Configuring caching" on page 248.

Upgrading and cache configuration

The Public Cloud Gateway introduces a new caching system with this release.

For details about how to configure the cache, see "Configuring caching” on page 248.

The existing cacheCounter section in the config.json is deprecated and no longer active. The following new sections control the caching behavior of the Public Cloud Gateway:

- cacheTimeout
- QuotaTimeouts

The cacheTimeout section is the direct replacement for the cacheCounter section in the previous release. While the cacheCounter semantic refreshed the cache after a given number of queries, the cacheTimeout replaces this semantic by a timeout value in seconds. In addition, in the current release, the caches are more precise and are invalidated if the related resource type for a project is modified, that is, created, deleted, or changed.

You can use the provided values to start, and then adjust them based on the workload run via the Public Cloud Gateway.

Upgrading Amazon EC2 multi-tenancy

The Public Cloud Gateway has some new capability for multi-tenancy for Amazon EC2.

The following resource types in Amazon EC2 are tagged with additional properties to provide a segregated view on a per project base:

- Virtual machine instances
- Volumes

The following tags are added:

**TenantUUID**

Describes the project to which the virtual machine belongs. The tag must contain a valid project UUID found in the identity view in the Administration user interface.

**RequestUserUUID**

Describes the user to which the virtual machine belongs. The tag must contain a valid user UUID found in the identity view in the Administration user interface.
Note: You must set at least the TenantUUID tag so that the virtual machine appears in OpenStack. The same is valid for existing volumes.

The upgrade is done by adding the above tags to the exiting resources in Amazon EC2 in the Amazon Management UI. As a result, the instances and volumes appear in the related projects.

For keypair, the upgrade is done differently because keypairs do not support tags. The _<project uuid>_ is a suffix to the name of the keypair. The upgrade of keypairs is done by renaming the related keypair in the Amazon Management UI.

Note: Existing keypairs cannot be added to IBM Cloud Orchestrator V2.4 so that they can be used in the “Deploying a virtual machine” on page 308 offering. They have to be reimported using the “Registering a key pair” on page 315 offering.

Performing post-upgrade configuration tasks
You must perform some configuration tasks after upgrading.

Because SoftLayer was not supported in SmartCloud Orchestrator V2.3, after upgrading the Public Cloud Gateway to V2.4, you must perform some configuration tasks if you are using SoftLayer.

Complete the following steps:
1. Edit /opt/ibm/pcg/etc/flavors.json adding the following Softlayer section:

```
"softlayer_default": {
  "SL.micro": {"name": "Micro", "cpu": 1, "ram": 1024, "disk": 80},
  "SL.small": {"name": "Small", "cpu": 1, "ram": 2048, "disk": 150},
  "SL.medium": {"name": "Medium", "cpu": 1, "ram": 4096, "disk": 400},
  "SL.large": {"name": "Large", "cpu": 2, "ram": 8192, "disk": 840},
  "SL.xlarge": {"name": "Extra Large", "cpu": 2, "ram": 8192, "disk": 840}
}
```

The final `flavors.json` looks like this:

```
{
  "default": {
    "m1small": {"name": "Small", "cpu": 1, "ram": 1024, "disk": 150},
    "m1medium": {"name": "Medium", "cpu": 2, "ram": 4096, "disk": 400},
    "m1large": {"name": "Large", "cpu": 4, "ram": 8192, "disk": 840},
    "m1xlarge": {"name": "Extra Large", "cpu": 8, "ram": 16384, "disk": 1680},
    "c1xlarge": {"name": "High-CPU Extra Large", "cpu": 20, "ram": 8192, "disk": 1680},
    "m2xlarge": {"name": "High-Memory Extra Large", "cpu": 6, "ram": 16384, "disk": 410}
  },
  "nioRegion1": {
    "m1small": {"name": "Small", "cpu": 1, "ram": 1024, "disk": 150},
    "m1medium": {"name": "Medium", "cpu": 2, "ram": 4096, "disk": 400},
    "m1large": {"name": "Large", "cpu": 4, "ram": 8192, "disk": 840},
    "m1xlarge": {"name": "Extra Large", "cpu": 8, "ram": 16384, "disk": 1680}
  },
  "softlayer_default": {
    "SL.micro": {"name": "Micro", "cpu": 1, "ram": 1024, "disk": 80},
    "SL.small": {"name": "Small", "cpu": 1, "ram": 2048, "disk": 150},
    "SL.medium": {"name": "Medium", "cpu": 1, "ram": 4096, "disk": 400},
    "SL.large": {"name": "Large", "cpu": 2, "ram": 8192, "disk": 840},
    "SL.xlarge": {"name": "Extra Large", "cpu": 2, "ram": 8192, "disk": 840}
  }
}
```


Common configuration tasks

There are some configuration tasks that you must perform.

Prerequisites

Before configuring the Public Cloud Gateway you must ensure that the requirements are satisfied.

General requirements

An Amazon Web Service (AWS) account with Amazon EC2 credentials for each tenant/project using the Public Cloud Gateway functionality is required. See the AWS Management Console for more information: https://console.aws.amazon.com/console/home

Setup an account in SoftLayer and create one or more users IDs. Each ID has its own unique password and API access key. The API access key is required to configure SoftLayer integration in the Public Cloud Gateway.

Network requirements

- **Port requirements** – The Public Cloud Gateway requires access to a number of ports in the installation environment and in the default Amazon EC2 security group. If these ports are blocked by a firewall or used by another process, some Public Cloud Gateway functions will not work.

<table>
<thead>
<tr>
<th>Port</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>22 TCP Outbound</td>
<td>SSH communication with the Virtual Machine instances.</td>
</tr>
<tr>
<td>ICMP</td>
<td>ICMP communication with the Virtual Machine instances.</td>
</tr>
<tr>
<td>443 TCP Outbound</td>
<td>HTTPS communication with: • Amazon EC2 management endpoints.</td>
</tr>
</tbody>
</table>

**Note:** Make sure that the Amazon EC2 security groups are configured according to the table. For more information about Amazon EC2 security groups, see [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html)

- **Nameserver (DNS) requirements** – Ensure that the Nameserver is configured correctly in both Central Server 2 and Central Server 3. Both Central Server 2 and Central Server 3 instances must be able to resolve the Amazon EC2 management endpoints as defined in /opt/ibm/pcg/etc/config.json. If the Nameserver is not configured correctly, Public Cloud Gateway functions will not work.

Access to public cloud resources

To provision virtual machines or use any services in the public cloud, users are required to have credentials to access public cloud resources. These credentials are then used in configuration of the Public Cloud Gateway.

Images in public cloud

To deploy patterns in public cloud, users are required to provide
pattern-specific private images or image templates in public cloud image repositories. See "Creating a supported image."

**Creating a supported image**

You can create a supported image.

The Public Cloud Gateway does not support Windows operating systems.

There are two major scenarios for Linux operating systems:

- "Creating cloud-init images for nova deployment for Linux"
- "Creating Workload Deployer cloud images for Linux" on page 242

**Creating cloud-init images for nova deployment for Linux:**

You can create cloud-init images for nova deployment for Linux.

The image creation is dependent on the target IaaS cloud.

**Amazon AWS EC2 images**

- Many of the actual Amazon AWS EC2 images are cloud-init enabled.
- If you are using an image that is not cloud-init enabled, follow the steps to add cloud-init support here: "Adding cloud-init to Linux images" on page 326.
- All Amazon AWS EC2 image for Linux provide cloud-init support natively.
- You must make a private copy out of the existing Amazon EC2 images.
  - Only private images can be used together with Public Cloud Gateway.
  - Follow the description in the Amazon EC2 documentation here: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami-ebs.html
- To enable password base authentication, see "Password authentication on Amazon EC2 images" on page 243.

**IBM SoftLayer**

To create a cloud-init-ready image template in SoftLayer, complete the following steps:

1. From the SoftLayer portal, create an instance SoftLayer-provided base OS images or create an instance from an existing image template that needs a cloud-init script installed.
2. To add cloud-init support to the image, follow the procedure in "Adding cloud-init to Linux images" on page 326.
3. When the instance is running, from the SoftLayer portal, access the computing instance: Device > Device List > Device Name.
4. From the Device List, select the Computing Instance from actions menu. Select Create Image Template.
5. Follow the prompts to create the image template.

**Non-IBM OpenStack**

Follow the OpenStack descriptions for OpenStack Linux image requirements. See http://docs.openstack.org/image-guide/content/ch_openstack_images.html Ensure that you have added the cloud-init support to the image.
Creating Workload Deployer cloud images for Linux:

You can create cloud-init images for Workload Deployer deployment for Linux.

The image creation is dependent on the target IaaS cloud.

**Note:** For Workload Deployer deployment, cloud-init version 0.6.0 or later is required for cloud images.

**Amazon AWS EC2 cloud-init images**

- Complete the procedure: [“Creating cloud-init images for nova deployment for Linux” on page 241](#) for Amazon AWS EC2 images.
- To enable password base authentication, see [“Password authentication on Amazon EC2 images” on page 243](#)

**Amazon AWS EC2 scp-init images**

Complete the procedure: [“Creating cloud-init images for nova deployment for Linux” on page 241](#) for Amazon AWS EC2 images.

Before you capture the image from the provisioned instance, update the instance by completing these steps:

1. Ensure that the following packages are installed in the instance:
   - [“Software prerequisites for Linux images (KVM or VMware hypervisors)” on page 330](#).
2. Create a supported image on Amazon EC2 for use with IBM Cloud Orchestrator: [“Enable Amazon EC2 image for scp-init activation” on page 243](#).

To enable password base authentication, see [“Password authentication on Amazon EC2 images” on page 243](#)

**IBM SoftLayer**

To create a cloud-init-ready image template in SoftLayer, complete the following steps:

1. From the SoftLayer portal, create an instance from SoftLayer-provided base OS images or create an instance from an existing image template that needs a cloud-init script installed.
2. To add cloud-init support to the image, follow the procedure in [“Adding cloud-init to Linux images” on page 326](#).
3. Perform steps 3 - 5 in [“Enable Amazon EC2 image for scp-init activation” on page 243](#).
4. When the instance is running, from the SoftLayer portal, access the computing instance: **Device > Device List > Device Name**.
5. From the Device List, select the Computing Instance from actions menu. Select **Create Image Template**.
6. Follow the prompts to create the image template.

**Non-IBM OpenStack**

Follow the OpenStack descriptions for OpenStack Linux image requirements. See [http://docs.openstack.org/image-guide/content/ch_openstack_images.html](http://docs.openstack.org/image-guide/content/ch_openstack_images.html) Ensure that you have added the scp-init support to the image.
Enable Amazon EC2 image for scp-init activation:

Enable root login to Amazon EC2 images

By default, some Amazon EC2 images do not allow you to log in as the root user and only allow you to log in using the ec2-user user. To log in as the root user, complete the following steps:

1. Log in to the image by ssh using the ec2-user user and run the following commands:
   ```bash
   sudo cp ~/.ssh/authorized_keys /root/.ssh/authorized_keys
   and
   sudo service sshd restart
   ```
2. Create a supported image on Amazon EC2 for use with IBM Cloud Orchestrator.

   **Note:** The following instructions are applicable for Linux images only. Windows images are not supported.

3. Download the following file to the Linux image: http://<Workload_Deployer_machine>/downloads/cloud-init/scp-cloud-init where <Workload_Deployer_machine> is the IP address of the server where the Workload Deployer component is installed.
4. Run the following commands as the root user:
   ```bash
   chmod 755 scp-cloud-init
   and
   ./scp-cloud-init install
   ```
5. Create the ovf-env.done file by using the following commands:
   ```bash
   mkdir -p /opt/IBM/AE/AR/
   and
   touch /opt/IBM/AE/AR/ovf-env.done
   ```
6. Delete the content of the /etc/rc.d/rc.local file. rc.local is an init script file provided by Amazon to edit the authentication_keys and sshd_config files. You must delete the content of the rc.local file to keep your preferences.
7. Select the Create Image option from the Amazon Web Service (AWS) Management Console to create an image from the instance. For more information about the AWS Management Console, see [https://console.aws.amazon.com/console/home](https://console.aws.amazon.com/console/home)

Password authentication on Amazon EC2 images:

You can allow password authentication and root login by using password authentication on Amazon EC2 images.

Log on to the Amazon EC2 image by using SSH and complete the following steps:

1. Edit the following file: /etc/ssh/sshd_config
2. Update the line that contains PasswordAuthentication:
   ```bash
   PasswordAuthentication yes
   ```
3. Update the line that contains PermitRootLogin:
   ```bash
   PermitRootLogin yes
   ```
4. Save the file.
5. Run the following command:
   ```
   sudo service sshd restart
   ```

**Tagging images with the configuration strategy:**

After a image is created, you must set the configuration strategy for an image, either on the image or at region level.

The configuration strategy describes if the image has a cloud-init or scp-init activation style configuration.

The target environments, Amazon EC2, SoftLayer and non-IBM-OpenStack (NIOS) have different capabilities to attach the configuration strategy on images.

These are the possibilities for image configuration strategy:

- **cloud-init**
  - New type introduced with IBM Cloud Orchestrator V2.4.

- **scp-init**
  - Existing type supported in SmartCloud Orchestrator V2.3.

If the image configuration strategy is not defined on the image, the image returns the configuration strategy as defined in the `ImageType` property in the `config.json` file. For more information, see [Table 31 on page 258](#).

If the image has a configuration strategy defined and the region has an `ImageType` property, the configuration strategy on the image has precedence. If the image does not have a configuration strategy defined and the region has an `ImageType` property, the image returns the `ImageType` value defined on the region level. If the region do not have an `ImageType` property and a configuration strategy is not defined on the image, then `scp-init` is used by default.

To define the configuration strategy on an image, see the following procedure depending on the target environment:

**Amazon EC2**

Amazon EC2 has the capability to apply tags to images. A tag is a key/value pair which is stored with the image meta data. To control the configuration strategy in Amazon EC2, there is a tag named `ImageType` which can have a value of `cloud-init` or `scp-init`.

**SoftLayer**

Configuration strategy is defined as part of the region configuration in the `config.json` file using the `ImageType` property on the region definition. In addition, SoftLayer allows setting a tag on the image itself using the SoftLayer management portal. Possible tag values are `cloud-init` or `scp-init`.

As an alternative to tagging images using the SoftLayer management portal, you can tag images using the following REST API call (on one line):

```bash
```

**Note:**

---

244  IBM Cloud Orchestrator 2.4: User's Guide
• SoftLayer does not support to set tag value during image template creation. It first requires creating the template and then editing the template by using templateID.
• In SoftLayer, tags on images only work within the account where the image resides. If an image is shared across accounts the image tag is not visible in the consuming accounts. Therefore, in the consuming accounts, an ImageType property in the region definition must be used or the image must be copied to the target account.

Non IBM OpenStack (NIOS)

For NIOS it is not supported to configure configuration on a per image base. Configuration strategy is defined as part of the region configuration in the config.json file.

Region level ImageType definition

The Public Cloud Gateway supports an ImageType property on the region definition. The ImageType property controls the configuration strategy returned for the images within this region. The ImageType property works as a default value for all images which do not have a defined configuration strategy.

Configuring flavors

There are limitations to the functionality provided by the Amazon EC2 API. One of the limitations is that there is no support for managing the flavors of the compute instances.

OpenStack API requires flavors for provisioning virtual machine. IBM Cloud Orchestrator must be able to return a flavor list for each region. The Public Cloud Gateway stores the current list of know flavors in the flavors.json file on Central Server 2 in the /opt/ibm/pcg/etc directory.

The following capabilities are supported:
• A global flavor list provided by the default section, if no remote cloud or region specific information is provided.
• A remote cloud type specific default flavor provided via the following sections:
  – ec2_default
  – nios_default
  – softlayer_default
• A region-specific flavor list provided via a section that has a section name as this region is defined in the config.json file. The name of region and name of the flavor section must match.

All the flavor definitions within flavors.json file must be valid for the related remote cloud regions. All definitions in these sections are snippets and provide configuration examples.

Note:
• Changes to the flavors.json file are only active after restarting the Public Cloud Gateway.
• Failure to the above assumptions result in provisioning errors.
• A flavor must not be removed if virtual machines with this flavor exist.
Amazon AWS EC2

Amazon AWS EC2 only supports a predefined list of flavors published on their website. The Public Cloud Gateway supplies a current list in the `flavor.json` under the `ec2_default` section. This hardcoded list can be extended or corrected based on the changes Amazon AWS EC2 provides. See [http://aws.amazon.com/ec2/instance-types/](http://aws.amazon.com/ec2/instance-types/).

The following rules apply for Amazon AWS EC2:

- You can modify the global list supplied in the `ec2_default` section. This affects all Amazon AWS EC2 regions except the ones that have a separate named section.
- You can add a new section for the specific Amazon AWS EC2 region with the name because this region is defined in the `config.json` file.

**Note:** With the Amazon AWS EC2 API, you cannot query or manage the supported flavors. Any changes to the list of flavors for Amazon AWS EC2 must match the published list on their website or the list of flavors show in the Amazon AWS EC2 management UI. Otherwise, the virtual machine has provisioning errors.

Non-IBM OpenStack

Non-IBM OpenStack is managed with the OpenStack EC2 provider. You cannot query or manage flavors. A manual step is required to synch and match the flavors available in each non-IBM supplied OpenStack region with the `flavors.json` file. If the detail of the flavors is later changed, the file must be updated again and the Public Cloud Gateway must be restarted.

The following rules apply for non-IBM OpenStack:

- You can modify the global list supplied in the `nios_default` section. This affects all non-IBM OpenStack regions except the ones that have a separate named section.
- You can add a new section for the specific non-IBM OpenStack region with the name because this region is defined in the `config.json` file.

1. To obtain the list of flavors that are supported on a particular non-IBM supplied OpenStack region, log on to a Nova node in that region and issue the command:
   ```
   nova flavor-list
   ```
   The following is an example of the response:

   ```
   root@nio3:~# nova flavor-list
   OS Password: 
   +----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-------------+
   | ID | Name | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public | extra_specs |
   +----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-------------+
   | 1 | m1.small | 2048 | 20 | 0 | | 1 | 1.0 | True | {} |
   | 2 | m1.medium | 4096 | 40 | 0 | | 2 | 1.0 | True | {} |
   | 3 | m1.large | 8192 | 80 | 0 | | 4 | 1.0 | True | {} |
   | 4 | m1.xlarge | 16384 | 160 | 0 | | 8 | 1.0 | True | {} |
   +----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-------------+
   ```

2. Edit the file `/opt/ibm/pcg/etc/flavors.json` file on the Central Server 2 node and use the data from the Name, Memory MB, Disk, and VCPUs columns to populate the name, RAM, disk, and CPU fields. Different flavors can be defined for each non-IBM supplied OpenStack and SoftLayer region. The region names must exactly match those defined in the `config.json` file.

3. There are default sections in this file where you can provide global defaults. The following possibilities are available:
default
The global default independent of the cloud type: SoftLayer or non-IBM OpenStack.

nios_default
A global default for non-IBM OpenStack regions.

softlayer_default
A global default for SoftLayer regions.

4. If the output in the previous step was generated in a non-IBM supplied OpenStack region that is defined as nioRegion1 in the config.json file, the resulting flavors.json would look similar to this:

```
"nioRegion1": {
    "m1.small": {"name":"Small", "cpu":1, "ram":2048, "disk":20},
    "m1.medium": {"name":"Medium", "cpu":2, "ram":4096, "disk":40},
    "m1.large": {"name":"Large", "cpu":4, "ram":8192, "disk":80},
    "m1.xlarge": {"name":"Extra Large", "cpu":8, "ram":16384, "disk":160},
}
```

Using this example, when accessing nioRegion1, IBM Cloud Orchestrator offers a list of four flavors, but any other non-IBM supplied OpenStack regions offers the six flavors that are defined for the default region.

Note:
• The ID in the flavors.json file, m1.small, must match the flavor name on non-IBM supplied OpenStack and not the flavor ID.
• IBM Cloud Orchestrator requires at least 512 MB of memory to be defined in flavors.

IBM SoftLayer

SoftLayer does not support flavors for deployment. The flavors.json defines the set of flavors you can use during deployment. SoftLayer only supports a certain list of possible values for CPU, RAM, and disk. These values can change over time. The possible values are visible if you try to create a cloud compute instance via the SoftLayer provided management UI. Only these values can be used for flavor definitions. If other values are used, you might receive deployment errors.

The following rules apply for IBM SoftLayer:
• You can modify the global list supplied in softlayer_default section. This affects all IBM SoftLayer regions except the ones that have a separate named section.
• You can add a new section for the specific IBM SoftLayer region with the name because this region is defined in the config.json file.

Note:
• IBM Cloud Orchestrator requires at least 512 MB of memory to be defined in flavors.
• SoftLayer provides a predefined set of values for CPU, RAM, and disk. Check the possible values in the SoftLayer documentation or in the SoftLayer management portal.
Configuring quotas
You can configure quotas.

Configuring default quota support

There are two types of quota definitions in the Public Cloud Gateway:

- A default quota set that is used if no project level quotas are defined
- Project specific quotas

Quotas are on a per project per region level.

The default quota set is stored within the config.json file located in the /etc subdirectory under the IBM Cloud Orchestrator installation directory. The default: /opt/ibm/pcg.

The file is in JSON format. This is the quota section of the file:

```json
{  
  "defaultQuota":{  
    "instances":"100",  
    "cores":"100",  
    "ram":"262144",  
    "gigabytes":"512",  
    "volumes":"2048",  
    "key_pairs":"100"  
  }  
}
```

To modify the config.json file, as root, open it in a text editor and change the values:

1. Connect to Central Server 2 via SSH. Default location: /opt/ibm/pcg/etc/config.json
2. Restart the Public Cloud Gateway by submitting the following command as root on the Central Server 2 command line: service pcg restart

Configuring project quota

Project quotas are managed via the Administration UI (Horizon). For information, see:

- "Editing the domain quotas" on page 220
- "Configuring project quotas" on page 225

**Note:** Project quotas cannot be deleted from the Administration UI. They can only be created and modified. The Public Cloud Gateway only supports a subset of the quotas as described in the "Quota support overview" on page 237.

Configuring caching

There are two different areas of caching in the Public Cloud Gateway: caching of resources and caching of quota actuals.

**Note:** The previous capability of cacheCounters in config.json is deprecated and no longer supported. See "Upgrading from SmartCloud Orchestrator V2.3 or V2.3.0.1" on page 152.

Both are configured within the config.json files in different sections.

Caching is required as the remote clouds have denial of service and API rate limits.
Without caching in public cloud gateway the remote clouds would disable the accounts after a short period of time, because there is a violation against the denial of service or against API rate limits.

The cache values describe when the public cloud internal caches are refreshed if no modifying resource requests are performed. Modifying resources requests are create / modify / delete style requests.

A modifying request would invalidate the caches for the triggering URL (region / project).

The values should be adapted so that the least amount of API calls are done against the remote clouds without impacting the responsiveness of the Public Cloud Gateway.

Caching of resources:

```json
{
  "cacheTimeout": {
    "serversTimeout": "180",
    "glanceImagesTimeout": "180",
    "availabilityZoneTimeout": "180",
    "volumesTimeout": "180",
    "keypairTimeout": "180"
  }
}
```

All values are in seconds.

- **serversTimeout**
  Defines the cache refresh interval for virtual machine instance related data.

- **glanceImagesTimeout**
  Defines the cache refresh interval for image related data. For example, if you add a new image to the IaaS, this is the time until it shows up in a "glance image-list" for that region.

- **availabilityZoneTimeout**
  Defines the cache refresh interval for changes related to availability-zones. Changes are normally infrequent because a new datacenter is added by the IaaS provider or a new region is defined in NIOS.

- **volumesTimeout**
  Defines the cache refresh interval for volume related data.

- **keypairTimeout**
  Defines the cache refresh interval for key pair related data.

Caching of quota actuals:

```json
{
  "QuotaTimeouts": {
    "quotaTimeout": {
      "serverQuotaTimeout": "600",
      "volumeQuotaTimeout": "600",
      "keypairQuotaTimeout": "600"
    }
  }
}
```

All values are in seconds. Each entry defines a refresh interval for the quota calculation:
serverQuotaTimeout
Defines the refresh interval in seconds for virtual machine instance related quota elements.

volumeQuotaTimeout
Defines the refresh interval in seconds for volume related quota elements.

keypairQuotaTimeout
Defines the refresh interval in seconds for key pair related quota elements.

Changing keystone administrator password
You can change the keystone administrator password.

Keystone administrator credentials are stored in /opt/ibm/pcg/etc/admin.json.

If you change the password for the user that is configured in the admin.json in OpenStack, you must also change the admin.json file in the Public Cloud Gateway etc directory.

Note: The values in the admin.json file are populated by default during the installation of IBM Cloud Orchestrator.

Go to the /opt/ibm/pcg/etc directory and open the admin.json file.

```json
{
    "auth": {
        "passwordCredentials": {
            "username": "xxx",
            "password": "xxx",
        },
        "tenantName": "xxx",
        "domainName": "xxx"
    }
}
```

The password is the keystone administrator password. This value must be encoded by using the encryptpassword.sh script that is available in the /opt/ibm/pcg directory. For more information about the encryptPassword.sh script, see "Command-line interface scripts."

After you change the password, restart the Public Cloud Gateway.

Changing a region name
You can change a region name.

Go to the /opt/ibm/pcg/etc directory and open the config.json property file. Replace the old name with new one.

For example, to show the change of the region name from EC2-001 to EC2region for an ec2 region. The original EC2 region is:

```
"ec2": [
    {
        "name": "EC2-001",
        "url": "https://ec2.us-east-1.amazonaws.com/",
        "enabled": true
    }
]
```

Change the region name:
Configure IBM Cloud Orchestrator for the new region and remove the entry for the old region. For more information about configuring a multiple region environment, see the "Installing a multiple-region environment."

Restart the Public Cloud Gateway by using the service pcg restart command. For more information about starting the Public Cloud Gateway, see the related CLI topic.

Delete the services of the old region from keystone:

```bash
[root@central-server-2 ~]# source ~/keystonerc
[root@central-server-2 ~]# keystone endpoint-list
WARNING: Bypassing authentication using a token and endpoint (authentication credentials are being ignored).
```

Delete all the endpoints related for the old region EC2-001 with the following example command:

```bash
Delete all the endpoints related for the old region EC2-001 with the following example command:
```

Chapter 3. Administering 251
[root@central-server-2 ~]# keystone endpoint-delete 0ff9e584b3d04c56af32e7b43ad5324d

**Restarting the Public Cloud Gateway**

You might need to restart the Public Cloud Gateway.

The Public Cloud Gateway runs as a service on Central Server 2. Some configuration tasks require a restart of the Public Cloud Gateway to activate changes:

- Changing a region name
- Changing a keystone administrator password
- Configuring caching
- Configuring default quotas
- Changing flavors
- Changing region settings

To restart the Public Cloud Gateway, complete the following steps:

1. Log on to Central Server 2 as root via SSH.
2. Check if the Public Cloud Gateway is running by executing as root:
   
   ```
   service pcg status
   ```

3. Restart the Public Cloud Gateway by executing as root:

   ```
   service pcg restart
   ```

4. Check the log of the Public Cloud Gateway for any errors and exceptions:

   ```
   less /var/log/pcg/pcg.log
   ```

**Managing Amazon EC2 using the Public Cloud Gateway**

The Public Cloud Gateway is not preconfigured for use with Amazon Elastic Compute Cloud (Amazon EC2) as part of the IBM Cloud Orchestrator. You must complete certain configuration tasks before using the Public Cloud Gateway.

1. Familiarize yourself with the Public Cloud Gateway. See “Public Cloud Gateway overview” on page 232.
2. Check that prerequisites are met. See “Prerequisites” on page 240.
3. Configure the Public Cloud Gateway for Amazon EC2. See “Configuring Public Cloud Gateway for Amazon EC2.”
4. Create a supported image. See “Creating a supported image” on page 241.
5. Configure quotas. See “Configuring quotas” on page 248.

For information about post-configuration steps, see “Performing post-configuration tasks” on page 266.

**Configuring Public Cloud Gateway for Amazon EC2**

You can configure the Public Cloud Gateway for Amazon EC2.

There are some configuration steps required in the following files to add a region and configure the credentials for a project:

- config.json
- credentials.json

**Note:** The examples shown in this section are only pieces of the config.json and credentials.json required for Amazon EC2-specific configurations. Both files contain additional sections you must not modify or delete as part of the Amazon EC2 configuration.
Configure regions in config.json: Go to the /opt.ibm/pcg/etc directory and open the config.json file.

The following code of the config.json is relevant for the Amazon EC2 region configuration:

```json
{
    "vcenters":{
        "ec2":{
            "name":"EC2-US-EAST-NORTHERN-VIRGINIA",
            "url":"https://ec2.us-east-1.amazonaws.com",
            "enabled":true
        },
        "name":"EC2-US-WEST-OREGON",
        "url":"https://ec2.us-west-2.amazonaws.com",
        "enabled":false
    },
    "name":"EC2-US-WEST-NORTHERN-CA",
    "url":"https://ec2.us-west-1.amazonaws.com",
    "enabled":false
},
    "name":"EC2-EU-IRELAND",
    "url":"https://ec2.eu-west-1.amazonaws.com",
    "enabled":false
},
    "name":"EC2-AP-SINGAPORE",
    "url":"https://ec2.ap-southeast-1.amazonaws.com",
    "enabled":false
},
    "name":"EC2-AP-TOKYO",
    "url":"https://ec2.ap-northeast-1.amazonaws.com",
    "enabled":false
},
    "name":"EC2-AP-SYDNEY",
    "url":"https://ec2.ap-southeast-2.amazonaws.com",
    "enabled":false
},
    "name":"EC2-SA-SAOPAULO",
    "url":"https://ec2.sa-east-1.amazonaws.com",
    "enabled":false
]
}
```

The cloud region configuration is described in the vCenters section. Each region is specified using three key-value pairs: name, url, and enabled.

The parameters in the config.json are explained in the following table. Update the enabled parameter to true if you want to specify that a particular region must be created in Keystone.
Table 27.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the region as it appears in keystone.</td>
</tr>
<tr>
<td>url</td>
<td>Amazon EC2: The URL of the Amazon EC2 vCenter that must be associated with the region. Default Amazon EC2 endpoints are defined. The data center is part of the URL. <a href="https://ec2.ap-southeast-2.amazonaws.com">https://ec2.ap-southeast-2.amazonaws.com</a>: ap-sourtheast-2 is the Amazon Data center.</td>
</tr>
<tr>
<td>enabled</td>
<td>Amazon EC2: Set to true if that datacenter is to be made available to the users of IBM Cloud Orchestrator. Set to false if that datacenter is not available. Do not use quotation marks.</td>
</tr>
</tbody>
</table>

Note: You must add a mapping for the project of the cloud administrator to the credentials.json. The default is "admin". If this entry is missing, you cannot add the availability zone to the domain via the IBM Cloud Orchestrator Administration UI.

```json
{
    "tenantName":"admin",
    "access_key_ID":"xxx",
    "secret_access_key":"xxx"
}
```

where xxx is a valid set of credentials to access your Amazon EC2 account.

Additional properties for Amazon EC2 on the region level. Example for SAOPAULO region:

```json
{
    "name":"EC2-SA-SAOPAULO",
    "url":"https://ec2.sa-east-1.amazonaws.com",
    "enabled":false / true,
    "ImageType": "cloud-init" or "scp-init",
}
```

Table 28.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageType</td>
<td>Defines on a region level what image activation type must be returned for images. The value is only used for images which are not already tagged with a image type. See &quot;Tagging images with the configuration strategy&quot; on page 244.</td>
</tr>
</tbody>
</table>
Configure cloud credentials in /opt/ibm/pcg/etc/credentials.json: This file is used to specify Amazon EC2 credentials for each project. For more information about defining projects, see Managing projects. The Amazon EC2 credentials are mapped to specific projects in IBM Cloud Orchestrator. These mappings are specified in the credentials.json configuration file.

Go to the /opt/ibm/pcg/etc directory and open the credentials.json file:

```json
{
    "cred":{
        "ec2":[
            {
                "tenantName":"demo",
                "access_key_ID":"xxx",
                "secret_access_key":"xxx"
            },
            {
                "tenantName":"admin",
                "access_key_ID":"xxx",
                "secret_access_key":"xxx"
            },
            {
                "tenantID":"xxxxxx",
                "access_key_ID":"xxx",
                "secret_access_key":"xxx"
            },
            {
                "tenantName":"*",
                "access_key_ID":"xxx",
                "secret_access_key":"xxx"
            }
        ]
    }
}
```

The parameters in the credentials.json are explained in the following table. Update these parameters if you want to specify credentials to project mappings and define what credentials must be used for the different projects specified.

**Table 29.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| tenantName      | tenantName specifies the OpenStack project entity, also known as tenant. The following options to identify a project exist:  
                |  • OpenStack project ID  
                |  • OpenStack project name  
                |  • Wildcard * to match any project.  
                | **Note:** Due to multi-domain support, a project name might not be unique, therefore a project ID must be used. To get the project ID, use the Administrative UI (Horizon) identity section. |
| access_key_ID   | This is the Amazon EC2 access key for the project.                           |
| secret_access_key | This is the Amazon EC2 secret access key used for the project. This value must be encoded using the encryptpassword.sh script that is available in the ../opt/ibm/pcg directory. |
Procedure to activate configuration changes:
1. Restart the Public Cloud Gateway by using the service pcg restart command. For more information, see “Command-line interface scripts” on page 272.
2. Execute the script refreshEndpoint.sh in the /opt/ibm/pcg directory to clean up caches related to region or endpoint information. See “Command-line interface scripts” on page 272.
3. Check the Public Cloud Gateway log in /var/log/pcg/pcg.log for problems.

Managing SoftLayer using the Public Cloud Gateway

The Public Cloud Gateway is not preconfigured for use with SoftLayer as part of IBM Cloud Orchestrator. You must perform certain configuration tasks before using the Public Cloud Gateway.

Before you begin

Familiarize yourself with the Public Cloud Gateway. See “Public Cloud Gateway overview” on page 232.

Procedure
1. Check that prerequisites are met. See “Prerequisites” on page 240.
2. You can integrate SoftLayer using the Public Cloud Gateway. See “Integrating SoftLayer using the Public Cloud Gateway.”
4. Create a supported image. See “Creating a supported image” on page 241.
5. Configure quotas. See “Configuring quotas” on page 248.

What to do next

For information about post-configuration steps, see “Performing post-configuration tasks” on page 266.

Integrating SoftLayer using the Public Cloud Gateway

You can integrate SoftLayer using the Public Cloud Gateway.

Before you begin

For general information about SoftLayer, see http://www.softlayer.com/.

Procedure
1. Setup an account in SoftLayer and create one or more users IDs. Each ID has its own unique password and API access key. The API access key is required in configuring SoftLayer integration in the Public Cloud Gateway.
2. Create IBM Cloud Orchestrator-ready images.
3. Set up the following configuration files as described in “Configuring the Public Cloud Gateway: Overview” on page 235. In particular, configure admin.json, config.json, credentials.json, and flavors.json.
4. Start or restart the Public Cloud Gateway.
Configuring the Public Cloud Gateway for SoftLayer

You can configure the Public Cloud Gateway for SoftLayer.

Before you begin

There are some configuration steps required in the following files to add a region and configure the credentials for a project.

- config.json
- credentials.json

Note: The examples shown in this section are only pieces of the config.json and credentials.json required for SoftLayer-specific configurations. Both files contain additional sections you must not modify or delete as part of the SoftLayer configuration.

Configure regions in config.json

Go to the /opt/ibm/pcg/etc directory and open the config.json file. The following piece of code of the config.json is relevant for the SoftLayer region configuration:

```json
{
    "vcenters": {
        "softlayer": [
            {
                "name": "SL-Dallas05",
                "dataCenter": "Dallas05",
                "url": "https://api.softwarelayer.com/",
                "enabled": true
            },
            {
                "name": "SL-Dallas06",
                "dataCenter": "Dallas06",
                "url": "https://api.softwarelayer.com/",
                "enabled": false
            },
            {
                "name": "SL-SanJose",
                "dataCenter": "SanJose",
                "url": "https://api.softwarelayer.com/",
                "enabled": false
            },
            {
                "name": "SL-Amsterdam",
                "dataCenter": "Amsterdam",
                "url": "https://api.softwarelayer.com/",
                "enabled": false
            },
            {
                "name": "SL-Seattle",
                "dataCenter": "Seattle",
                "url": "https://api.softwarelayer.com/",
                "enabled": false
            },
            {
                "name": "SL-WashingtonDC",
                "dataCenter": "WashingtonDC",
                "url": "https://api.softwarelayer.com/",
                "enabled": false
            },
            {
                "name": "SL-Singapore",
                "dataCenter": "Singapore",
                "url": "https://api.softwarelayer.com/",
                "enabled": false
            }
        ]
    }
}
```
The cloud region configuration is described in the vCenters section. Each region is specified using three key-value pairs: name, url, and enabled. The parameters in the config.json are explained in the following table. Update the enabled parameter to true if you want to specify that a particular region must be created in Keystone.

Table 30. Parameters that are used in the config.json file

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the region as it appears in keystone.</td>
</tr>
<tr>
<td>dataCenter</td>
<td>Name of the SoftLayer data center the region is connected to.</td>
</tr>
<tr>
<td>enabled</td>
<td>SoftLayer: Set to true if that datacenter is to be made available to the users of IBM Cloud Orchestrator. Set to false if that datacenter is not available. The value of &quot;enabled&quot; is either true or false: do not use quotation marks.</td>
</tr>
</tbody>
</table>

Configure cloud credentials in /opt/ibm/pcg/etc/credentials.json.

Additional properties available for SoftLayer regions. Example for Singapore datacenter:

```json
{
   "name": "SL-Singapore",
   "dataCenter": "Singapore",
   "url": "https://api.softlayer.com/",
   "enabled": false / true,
   "imageType": "cloud-init" or "scp-init",
   "privateNetworkOnly": false / true,
   "primaryVlanId": "600516",
   "backendVlanId": "600518"
}
```

Table 31. Parameters that are used in the config.json file

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageType</td>
<td>Defines on a region level what image activation type must be returned for images. The value is only used for images which are not already tagged with a image type. See &quot;Tagging images with the configuration strategy&quot; on page 244.</td>
</tr>
</tbody>
</table>
Table 31. Parameters that are used in the config.json file (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>privateNetworkOnly</td>
<td>If it is set to true, the virtual machine has a single private NIC (backend). If it is set to false, the virtual machine has a private (backend) and a public (primary) NIC. The default value is false.</td>
</tr>
<tr>
<td>primaryVlanId</td>
<td>VLAN ID that connects the virtual machine to the internet (public VLAN). VLAN ID is the SoftLayer resource ID that describes a VLAN. If primaryVlanId is not specified, the SoftLayer default is used. For the public network, it is critical to configure the firewall with the appropriate rules for the user ID. The Public Cloud Gateway executes the deployment with the configured user ID (as specified in the credentials.json file) so that the firewall rules as defined for that user are applied for the public network of the provisioned virtual machine (for example, only HTTP traffic, port 80 allowed).</td>
</tr>
<tr>
<td>backendVlanId</td>
<td>VLAN ID that connects the virtual machine to the management network (private VLAN). VLAN ID is the SoftLayer resource ID that describes a VLAN. If backendVlanId is not specified, the SoftLayer default is used.</td>
</tr>
</tbody>
</table>

Note: To obtain the correct VLAN ID, perform the following steps:
1. Log on to SoftLayer portal at [https://control.softlayer.com/](https://control.softlayer.com/).
3. Choose the VLAN that you want to use for provisioning and select it to open the VLAN details.
4. Copy the VLAN ID from the browser URL. For example, if the URL is [https://control.softlayer.com/network/vlans/600516](https://control.softlayer.com/network/vlans/600516) then 600516 is the correct ID. Do not confuse the VLAN ID with the VLAN number displayed on the web page.

Configure cloud credentials in /opt/ibm/pcg/etc/credentials.json

This file is used to specify SoftLayer credentials for each project. For more information about defining projects, see Managing projects. The SoftLayer credentials are mapped to specific projects in IBM Cloud Orchestrator. These mappings are specified in the credentials.json configuration file.

Go to the /opt/ibm/pcg/etc directory and open the credentials.json file:

```
{
  "cred":{
    "softlayer":[
      {
        "tenantName":"admin",
        "user_id":"xxx",
```
The parameters in the `credentials.json` are explained in the following table. Update these parameters if you want to specify credentials to project mappings and define what credentials must be used for the different projects specified.

### Table 32. Parameters that are used in the `credentials.json` file

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tenantName</code></td>
<td>Specifies the OpenStack project entity, also known as tenant. The following options to identify a project exist:</td>
</tr>
<tr>
<td></td>
<td>• OpenStack project ID</td>
</tr>
<tr>
<td></td>
<td>• OpenStack project name</td>
</tr>
<tr>
<td></td>
<td>• Wildcard * to match any project</td>
</tr>
<tr>
<td></td>
<td>Due to multi-domain support a project name might not be unique, therefore a project ID must be used. To get the project ID, use the</td>
</tr>
<tr>
<td></td>
<td>Administrative UI (Horizon) identity section.</td>
</tr>
<tr>
<td><code>user_id</code></td>
<td>This is the SoftLayer account user ID used for the project.</td>
</tr>
<tr>
<td><code>api_access_key</code></td>
<td>This is the SoftLayer api access key. This value must be encoded using the encryptpassword.sh script that is available in the ../opt/ibm/pcg directory.</td>
</tr>
</tbody>
</table>

**Note:** You must add a mapping for the project of the cloud administrator to the `credentials.json`. The default is `admin`. If this entry is missing, you cannot add the availability zone to the domain via the Administration user interface.

```json
{
  "tenantName":"admin",
  "user_id":"xxx",
  "api_access_key":"xxx"
}
```

where `xxx` is a valid set of credentials to access your SoftLayer account.

**Activating configuration changes**
**Procedure**

1. Restart the Public Cloud Gateway by using the `pcg restart` command. For more information about starting the Public Cloud Gateway, see "Command-line interface scripts" on page 272.

2. Execute the script in the `/opt/ibm/pcg/refreshEndpoint.sh` to clean up caches related to region or endpoint information.

3. Check the Public Cloud Gateway log in `/var/log/pcg/pcg.log` for problems.

**Managing non-IBM supplied OpenStack using the Public Cloud Gateway**

IBM Cloud Orchestrator is unable to manage non-IBM supplied OpenStack directly due to the differences between IBM supplied OpenStack and non-IBM supplied OpenStack. As a result of this, the Public Cloud Gateway can be used as an interface between IBM Cloud Orchestrator and non-IBM supplied OpenStack.

The Public Cloud Gateway provides a compatibility layer that enables IBM Cloud Orchestrator to manage Amazon EC2 images and instances by performing the necessary translation to the Amazon EC2 API. This functionality can be used to enable IBM Cloud Orchestrator to manage non-IBM supplied OpenStack services by accessing them through their Amazon EC2 API.

**Note:** Not all EC2 operations are supported by the OpenStack EC2 implementation. For example, the `Resize Instance` functionality is not supported in Public Cloud Gateway by the OpenStack EC2 API.

Additional information on the capabilities of the OpenStack EC2 implementation is available under API Feature Comparison at: [https://wiki.openstack.org/wiki/Main_Page](https://wiki.openstack.org/wiki/Main_Page).

The Public Cloud Gateway is not preconfigured for use with Amazon Elastic Compute Cloud (Amazon EC2) as part of IBM Cloud Orchestrator. You must complete certain configuration tasks before using the Public Cloud Gateway.

1. Familiarize yourself with the Public Cloud Gateway. See "Public Cloud Gateway overview" on page 232.

2. Check that prerequisites are met. See "Prerequisites" on page 240.

3. Configure the Public Cloud Gateway to manage non-IBM supplied OpenStack. See "Configure the Public Cloud Gateway to manage non-IBM supplied OpenStack" on page 262. You must already have one or more OpenStack regions that are configured and functioning. For information about how to install and configure a basic OpenStack instance, see [http://docs.openstack.org](http://docs.openstack.org).

4. Create a supported image. See "Creating a supported image" on page 241.


For information about post-configuration steps, see "Performing post-configuration tasks" on page 266.
Configure the Public Cloud Gateway to manage non-IBM supplied OpenStack

The following configuration topics assume that you already have one or more OpenStack regions that are configured and functioning. Instructions on how to install and configure a basic OpenStack instance can be found at [http://docs.openstack.org](http://docs.openstack.org).

Configure the Public Cloud Gateway regions for non-IBM supplied OpenStack:

The Public Cloud Gateway requires connection details for every non-IBM supplied OpenStack region it manages.

**Before you begin**

The examples in this section are only pieces of the `config.json` required for the non-IBM supplied OpenStack-specific configurations. The file contains additional sections that you must not modify or delete as part of the non-IBM supplied OpenStack configuration.

**About this task**

To obtain the connection details, log on to the host that is running the Keystone service in each non-IBM supplied OpenStack region and complete the following steps:

**Procedure**

1. Run the following command:
   ```
   keystone service-list
   ```
   and look for the entry for Amazon EC2 and take note of the ID.

2. Run the following command:
   ```
   keystone endpoint-list
   ```
   and find the entry where the `service_id` matches the Amazon EC2 ID from the previous step. Take a note of the `publicurl` also. It will be something similar to `http://<address>:8773/services/Cloud`

   **Note:** If you want the Public Cloud Gateway to manage more than one non-IBM supplied OpenStack region, repeat these steps on the Keystone server for each non-IBM supplied OpenStack region. This is required to obtain the Amazon EC2 API interface address for each region.

3. The Public Cloud Gateway reads the connection details at startup from the `/opt/ibm/pcg/etc/config.json` file on the Central Server 2 node. By default, this file only contains the details of the Amazon EC2 regions. This file must be updated to add the non-IBM supplied OpenStack regions to a data block tagged `nios` inside the `vcenters` scope similar to this partial example:

   ```
   "vcenters":{
     "nios":[
       {
         "name": "nioRegion1",
         "url": "http://172.16.108.12:8773/services/Cloud/",
         "enabled": true
       },
       {
         "name": "nioRegion2",
         "url": "http://172.16.108.13:8773/services/Cloud/"
       }
     ]
   }
   ```
Note: The url that is obtained from Keystone has a trailing “/” appended to it.

Note: The region name specified in the nios tag section must be a unique name for that particular region.

Activating configuration changes

4. To activate configuration changes, complete the following steps:
   a. Restart the Public Cloud Gateway by using the service pcg restart command.
   b. Execute the script refreshEndpoint.sh in the /opt/ibm/pcg directory to clean up caches related to region or endpoint information.

For more information, see “Command-line interface scripts” on page 272.

What to do next

If the connection details contain a host name rather than an IP address, make sure that the Central Server 2 node can resolve the host names and add entries to the /etc/hosts file if required. Alternatively, use the IP address rather than the host name in the config.json file.

Configure non-IBM supplied OpenStack EC2 credentials:

The credentials that are used on the Amazon EC2 API are different from the credentials that are used by the OpenStack API. As a result, you must generate these special credentials and configure the Public Cloud Gateway to use them when accessing the non-IBM supplied OpenStack services.

The credentials that the Public Cloud Gateway uses on the Amazon EC2 API interface are checked by the Keystone service that is running in the non-IBM supplied OpenStack region. As a result, you must use the non-IBM supplied OpenStack Keystone service to generate the Amazon EC2 API credentials.

1. The command that is used to create the Amazon EC2 credentials requires the 32-digit IDs of the user and tenant that are copied during the Amazon EC2 API calls. Log on to the non-IBM supplied OpenStack Keystone host in each non-IBM supplied OpenStack region and obtain these IDs using the commands:
   keystone user-list
   keystone tenant-list

   Note: The user and tenant that are chosen must have sufficient access to perform the necessary actions on Nova and Glance. Use the admin user and admin tenant unless you have suitable alternatives that are configured on the non-IBM supplied OpenStack Keystone.

2. Create the Amazon EC2 credentials using the command:
   keystone ec2-credentials-create --tenant-id <tenant ID> --user-id <user ID>

   If successful, the command returns two new 32-bit keys that are called Access and Secret.
Note: Amazon EC2 credentials that are created on non-IBM supplied OpenStack Keystone apply to both tenant and user when using these credentials. For example, when deploying a virtual machine, the virtual machine is created using the tenant and the user specified in the previous command.

3. The following example shows the previous steps where Amazon EC2 credentials are created on the non-IBM supplied OpenStack using the admin tenant and the admin user:

```
root@nio3:/etc/glance# keystone user-list
+----------------------------------+---------+---------+--------------------+
| id | name   | enabled | email             |
+----------------------------------+---------+---------+--------------------+
| 475e0cb45d1049cb5bdf1eb508b391 | admin   | True    | admin@domain.com  |
| 90165358e49460297bb33db8484cf  | cinder  | True    | cinder@domain.com |
| 79fabc0d2d8f34aabc0f7797a8000a | demo    | True    | demo@domain.com   |
| 00cbe0b67d734e7f99ca5d7077b0fc1 | glance  | True    | glance@domain.com |
| 10f54c9b123147c488ae3c942134a8c | nova    | True    | nova@domain.com   |
| 940f130d79cc46b9ae38ae6b9d929767 | quantum | True    | quantum@domain.com|
+----------------------------------+---------+---------+--------------------+
```

```
root@nio3:/etc/glance# keystone user-list
```

```
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>40a39220bf5747edaac54216b5e8eb60</td>
<td>admin</td>
<td>True</td>
</tr>
<tr>
<td>7cda91633a43d1973bdbe0b28b76</td>
<td>demo</td>
<td>True</td>
</tr>
<tr>
<td>0420552b5721451a9d4d5e96a79444</td>
<td>service</td>
<td>True</td>
</tr>
</tbody>
</table>
```

```
root@nio3:/etc/glance# keystone ec2-credentials-create
--tenant-id 40a39220bf5747edaac54216b5e8eb60
--user-id 475e0cb45d1049cb5bdf1eb508b391

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>access</td>
<td>7e3d858e92324564a31e5d9b50fa62f0</td>
</tr>
<tr>
<td>secret</td>
<td>9db7e415c00ae496a90bcb0f9d725</td>
</tr>
<tr>
<td>tenant_id</td>
<td>40a39220bf5747edaac54216b5e8eb60</td>
</tr>
<tr>
<td>user_id</td>
<td>475e0cb45d1049cb5bdf1eb508b391</td>
</tr>
</tbody>
</table>
```

Note: If the Keystone ec2-credentials-create command is run a second time, even if the same user ID and tenant ID are used, the result are different and the previous access and secret key becomes invalid. Also, ensure that the commands are run on each separate non-IBM supplied OpenStack region that is controlled by the Public Cloud Gateway. This is required as different Keystone instances generate different IDs for the same user name or tenant name.

4. Before the secret key can be used, it must be base-64 encoded using the encryptpassword.sh script:

- Edit /opt/ibm/pcg/etc/credentials.json on the Central Server 2 machine.
- In the nios section, insert a new block of data with the correct region name, tenant name, access key, and encoded secret key.

Note: Take care to insert the appropriate comma if a new block is being appended to an existing section.

Note: Credentials must be defined for all IBM Cloud Orchestrator tenants.

- Save the file.

Note: The examples in this section are only pieces of the credentials.json required for the non-IBM supplied OpenStack specific configurations. The file
contains additional sections that you must not modify or delete as part of the non-IBM supplied OpenStack configuration.

The following example of credentials.json shows the formatting:

```json
{
    "cred": {
        "nios": [
            {
                "tenantName": "admin",
                "access_key_ID": "7e3d858e92324564a31e5d9b50fa62f0",
                "secret_access_key": "OGQ3ZTE1YzBhZTY0WlZyTk1mNjZDhOWRiYjcyNQ==",
                "region": "nioRegion1"
            }
        ]
    }
}
```

The parameters in the credentials.json are described in the following table. Update these parameters if you want to specify credentials to project mappings and define what credentials must be used for the different projects specified.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| tenantName     | TenantName specifies the OpenStack project entity, also known as tenant. The following options to identify a project exist:  
• OpenStack project ID  
• OpenStack project name  
• Wildcard * to match any project.  
Note: Due to multi-domain support, a project name might not be unique, therefore a project ID must be used. To get the project ID, use the Administrative UI (Horizon) identity section. |
| access_key_ID  | This is the non-IBM supplied OpenStack access key for the project. |
| secret_access_key | This is the non-IBM supplied OpenStack secret_access_key used for the project. This value must be encoded using the encryptpassword.sh script that is available in the ../opt/ibm/pcg directory. |
| region         | This is the name of the region as defined in the config.json file for the non-IBM supplied OpenStack. This tenant mapping is valid. |

Note: You must add a mapping for the project of the cloud administrator to the credentials.json. The default is "admin". If this entry is missing, you cannot add the availability zone to the domain via the IBM Cloud Orchestrator Administration UI.

```json
{
    "tenantName": "admin",
    "access_key_ID": "xxx",
    "secret_access_key": "xxx"
}
```

where xxx is a valid set of credentials to access your Amazon EC2 account.
Activating configuration changes

Complete the following steps:
1. Restart the Public Cloud Gateway by using the `service pcg restart` command.
2. Execute the script `refreshEndpoint.sh` in the `/opt/ibm/pcg` directory to clean up caches related to region or endpoint information.

For more information, see "Command-line interface scripts" on page 272.

Performing post-configuration tasks

You must complete these post-configuration tasks after you configure the Public Cloud Gateway.

About this task

You can deploy a single virtual machine or a virtual pattern. The post-configuration steps are different for each scenario.

Procedure

1. For deployment using a single virtual machine, complete the following steps:
   a. Add the newly-defined Public Cloud Gateway managed region or availability zone to:
      Domain
      See "Assigning a zone to a domain" on page 219.
      Project
      See "Assigning a zone to a project" on page 224.
   b. Register a new SSH key for deployment. See "Registering a key pair" on page 315.
   c. If you want to use additional disks during deployment, you must create volumes for the project. You can create volumes using the OpenStack Cinder Storage Volumes toolkit.
   d. Add `cloud-init` to Linux operating system images, as described in "Adding cloud-init to Linux images" on page 326.
   e. Deploy the virtual machine as described in "Deploying a single virtual machine".

2. For deployment using virtual patterns, complete the following steps:
   a. Update the NTP servers list to include a valid NTP server for the new Public Cloud Gateway managed region. For more information about the NTP servers list, see "Setting NTP servers" on page 94.
   The default NTP configuration is only valid for new servers that can reach the Deployment Server node via DNS. A publicly accessible NTP server is required for servers in public clouds. Examples are:
      • `0.amazon.pool.ntp.org`
      • `1.amazon.pool.ntp.org`
      • `2.amazon.pool.ntp.org`
      • `3.amazon.pool.ntp.org`
   For SoftLayer, you can use `servertime.service.softlayer.com` as an NTP server.
b. Before you create a Linux image, ensure that the software requirements are met, as described in "Software prerequisites for Linux images (KVM or VMware hypervisors)" on page 330.

c. Deploy the virtual pattern as described in Chapter 7, “Managing and deploying virtual patterns,” on page 343.

Results

You can now deploy a virtual machine or virtual pattern by using the Public Cloud Gateway.

Troubleshooting

Use this section to resolve any issues that you may encounter when using the Public Cloud Gateway.

Logs are stored in the /var/log/pcg.log.

The Public Cloud Gateway uses log4j logging. The log4j.properties file is located in the /opt/ibm/pcg/etc directory.

For more information about the properties in the log4j.properties file, see the documentation on the Log4j web site: http://logging.apache.org/log4j

Loss of functionality in Public Cloud Gateway cloud groups

Loss of functionality may occur in Public Cloud Gateway cloud groups in IBM Cloud Orchestrator, where there has been heavy load on the Public Cloud Gateway cloud groups.

Symptom
As stated in description.

Cause
This is due to exceeding the Amazon Web Service (AWS) API Request Rate limit. The Public Cloud Gateway has addressed this issue by introducing a caching mechanism.

Solution
See “Configuring caching” on page 248.

Related tasks:

The Public Cloud Gateway is deployed as part of the IBM Cloud Orchestrator installation on the Central Server 2 machine. However, the Public Cloud Gateway is not enabled by default and certain updates to the configuration files are required before you use the Public Cloud Gateway.

Region names displayed incorrectly in the Virtual Image page

A known issue exists where IBM Cloud Orchestrator removes the name after an "_" in the region name when registering images.

Symptom
EC2-US-WEST_NORTHERN-CA and EC2-US-WEST_OREGON regions are displayed as EC2-US-WEST when registering an image. This error prevents you from selecting images from both regions.

Solution

1. Edit /opt/ibm/pcg/etc/config.json on Central Server 2 and replace "_" (underscore) in region names with a "-" (dash).
2. Restart the Public Cloud Gateway to allow the changes to take effect:
3. Log in to IBM Cloud Orchestrator and wait for the new regions to display. Once the images are displayed, delete the old EC2-US-WEST_NORTHERN-CA and EC2-US-WEST_OREGON cloud groups and hypervisors. You can also delete any registered images that belong to these regions.

4. Log in to the Central Server 2 and run the following command:
   ```bash
   keystone endpoint-list
   ```
   and identify the old endpoints by their region name.

5. Delete the old endpoints for EC2-US-WEST_NORTHERN-CA and EC2-US-WEST_OREGON, by running the following command:
   ```bash
   keystone endpoint-delete <endpoint-id>
   ```
   **Note:** Be careful not to delete any valid endpoints.

6. Images can now be registered for both regions.

Related tasks:
- "Configuring the Public Cloud Gateway: Overview" on page 235

Unable to connect to a public cloud due to missing credentials

In the Public Cloud Gateway, you might receive the error **Unable to connect to public cloud due to missing credentials.**

**Symptom**
As stated in description.

**Cause**
This is due to tenants or projects being present in IBM Cloud Orchestrator that are not accounted for in the credentials.json file.

**Solution**
You can resolve this problem in one of the following ways:
- Add credentials for each tenant in IBM Cloud Orchestrator to the credentials.json file.
- Add credentials for a specific tenant by ID in IBM Cloud Orchestrator to the credentials.json file.
- Add credentials, where the tenantName is * as stated in step 5 in the related configuring topic. This ensures that these credentials are applied to each tenant that is not explicitly stated in credentials.json file.

Related tasks:
- "Configuring the Public Cloud Gateway: Overview" on page 235

The Public Cloud Gateway is deployed as part of the IBM Cloud Orchestrator installation on the Central Server 2 machine. However, the Public Cloud Gateway is not enabled by default and certain updates to the configuration files are required before you use the Public Cloud Gateway.
Unable to deploy instance to non-IBM supplied OpenStack

Unable to deploy an instance to non-IBM supplied OpenStack with error:
"org.apache.http.impl.client.DefaultRequestDirector [WARN] Authentication error: Unable to respond to any of these challenges: {}"

While Public Cloud Gateway might appear to behave normally, the hypervisors are present, the cloud group exists, registering images is successful, the following is shown in the log file when deploying an instance to non-IBM supplied OpenStack.

```
[2013-08-01 09:00:33,607] org.apache.http.impl.client.DefaultRequestDirector [WARN] Authentication error: Unable to respond to any of these challenges: {}
```

```
Caused by: com.amazonaws.AmazonClientException: Unable to unmarshall error response
(Content is not allowed in prolog.)
```

```
Caused by: org.xml.sax.SAXParseException: Content is not allowed in prolog.
```

Solution:

The credentials used to connect to non-IBM supplied OpenStack have read but not write permissions. The Public Cloud Gateway requires credentials with permissions to deploy instances.

Inconsistencies in deployed instance names

When server instances are deployed on non-IBM supplied OpenStack, the names that are given to the instances in the non-IBM supplied OpenStack differ from the names that are given to the instances in IBM Cloud Orchestrator.

This is a known issue in the Amazon EC2 API implementation in OpenStack versions before the Grizzly release.

For more information about this known issue, see the following link:

[https://bugs.launchpad.net/nova/+bug/1096821](https://bugs.launchpad.net/nova/+bug/1096821)

MAC address field is empty

No information is displayed for the MAC address field when viewing non-IBM supplied OpenStack or Amazon EC2 virtual system instances.

**Symptom**

Once a non-IBM supplied OpenStack or Amazon EC2 virtual system instance is deployed, open the Virtual System Instances panel, by clicking PATTERNS > Instances > Virtual System Instances (Classic). You see that no information is displayed for the MAC address field.

**Solution**

This information is not available when using the Public Cloud Gateway.
Quota troubleshooting
Resolve any quota issues that you encounter when using the Public Cloud Gateway.

Default quota is not defined large enough
Without any customization, a default quota definition exists in the config.json. There are situations in which this default quota definition is too small.

Resolution: Either create a project level quota in the Quota tab of the Project page in the Administration user interface or increase the default quota definition in config.json.

Project quota definition is too small
Resolution: If a project level quota definition exists, the values can only be changed in the Quota tab of the Project page in the Administration user interface.

Existing virtual machine instances already consume more resources than the quota allows
Resolution: Count the number of instances, cores, RAM, and volume usage and update the corresponding quota values. Volumes is the sum of the VM instance volume and the additional disks. There is a gigabytes value in the quotas which defines the largest possible virtual machine instance possible. It is possible that you have reached this limit.

Too many key-pairs already exist
Resolution: Key pairs are stored on a per project base post-fixed with the project ID. Sum up all the key pairs that have the same project ID and adjust the quota definition for key pairs accordingly.

Too much storage is already consumed than is defined in the quota
Resolution: Volumes are the sum of the virtual machine instance volume and the additional disks. Adjust the quota definition for volumes accordingly.

Provisioning failed even though quota has not been reached yet
There might be situation where the capacity of the region (EC2, SoftLayer or NIOS) is already exhausted prior to the defined quotas.

Resolution:
• Check if you have set the quota for the region and projects higher than the capacity of the region. Either lower the quota for the related projects or increase the capacity of the region.
• Check if you have additional out-of-bound-created resources within the region. If so, make sure, using ACL, that these resources are not visible to the user ID you have configured for the region access in credentials.json.
Failure to generate admin token
The Public Cloud Gateway fails with Unable generate admin token error.

Problem

The Public Cloud Gateway startup fails with Unable generate admin token and a HybridUnauthorizedException errors.

During startup of Public Cloud Gateway an admin token is generated based on the following configuration information in the etc directory of the Public Cloud Gateway:
* admin.json
* config.json

Admin.json content:
```json
{
  "auth":{
    "passwordCredentials":{ 
      "username":"xxxx",
      "password":"yyyyy"
    },
    "tenantName":"zzzz",
    "domainName": "dddd"
  }
}
```

The username must be a user ID which has admin rights. The password must be encrypted via the encryptPassword.sh. The tenantName must be set to the tenant name of the admin user. The domainName is optional and defaults to the Default domain. Set domainName to the domain of the admin user.

Note: Required if the user is in a non-default domain.

Resolution

Make sure the values in admin.json match to your admin userid in the system.

Config.json content:
```json
"auth":{
  "provider":"keystone",
  "service_url":"http://KeystoneHost:5000",
  "admin_url":"http://KeystoneHost:35357"
}
```

If you manually changed the content of the service_url or the admin_url, the admin token cannot be created.

Make sure that the KeystoneHost is set to the host name where keystone is installed in your IBM Cloud Orchestrator environment. During installation the values are configured based on your topology selection.
Reference
This section provides reference information for the Public Cloud Gateway.

Key pairs
Key pairs are needed to access the virtual machines that you deploy in a pattern. When you deploy a virtual machine, these keys are injected into the instance to allow password-less SSH access to the instance.

The default key pair that is created from the Self-service user interface in Amazon EC2 regions is appended with the user ID of the user that created the key pair. For example, if the user creating the key pair in the Self-service user interface is admin, the name of the key pair that is created is Amazon EC2 is `default_admin`. For information about managing key pairs, see "Managing key pairs" on page 315.

Command-line interface scripts
Command Line Interface (CLI) scripts are available in the `/opt/ibm/pcg` directory. These scripts are used to perform manual tasks, for example, starting the Public Cloud Gateway, encrypting a password, changing port numbers and so on.

Note: The default port number is 9797. To change this, you must edit the `startServer.sh` script and change the value of `-DHybrid.Port` to the new port number.

- `encryptPassword <plain text password>` - writes encrypted password to stdout. `EncryptPassword.sh`. This script is located in the `/opt/ibm/pcg` directory. It is used for encrypting passwords and access keys used in admin.json and credentials.json files located in the `/opt/ibm/pcg/etc` directory. The command must be run in the directory where `encryptPassword.sh` script is located.
- `service pcg start` - used to start the Public Cloud Gateway server with the default settings.
- `service pcg stop` - used to stop the Public Cloud Gateway server.
- `service pcg restart` - used to restart the Public Cloud Gateway server.
- `refreshEndpoint.sh <admin userid> <admin password> <bpmhostname:port of bpm node>` - refreshes the IBM Cloud Orchestrator endpoint cache. This action is required if region changes are done within the Public Cloud Gateway `config.json` file.

Password authentication on Amazon EC2 images
You can allow password authentication and root login by using password authentication on Amazon EC2 images.

Log on to the Amazon EC2 image by using SSH and complete the following steps:
1. Edit the following file: `/etc/ssh/sshd_config`
2. Update the line that contains `PasswordAuthentication`:
   ```
   PasswordAuthentication yes
   ```
3. Update the line that contains `PermitRootLogin`:
   ```
   PermitRootLogin yes
   ```
4. Save the file.
5. Run the following command:
   ```
   sudo service sshd restart
   ```
### Configuring the Public Cloud Gateway regions to use a proxy server

You can configure the Public Cloud Gateway regions to use a proxy server.

#### About this task

You can specify proxy server details at region level or at vCenter level (ec2 or nios). The proxy configuration that you specify at region level overrides the proxy configuration specified at vCenter level.

The Public Cloud Gateway reads the proxy server details at startup from the /opt/ibm/pcg/etc/config.json file on the Central Server 2.

In the following example, the proxy server configuration is defined at region level for the EC2-EU-IRELAND region. Also a second proxy server configuration is defined for the nios vCenter.

```json
"vcenters":
{
  "ec2":
  {
    "name": "EC2-EU-IRELAND",
    "url": "https://ec2.eu-west-1.amazonaws.com",
    "enabled": true,
    "proxy":
    {
      "host": "proxy-server-host",
      "username": "admin",
      "password": "cGFzc3cwcmQ=",
      "port": 3128
    }
  },
  {
    "name": "EC2-US-WEST_OREGON",
    "url": "https://ec2.us-west-2.amazonaws.com",
    "enabled": true
  }
},

"nios":
{
  "nioRegion1",
  "url": "http://172.16.108.12:8773/services/Cloud/",
  "enabled": true
}

"proxy":
{
  "vcenter": "nios",
  "host": "172.17.41.129",
  "username": "admin",
  "password": "cGFzc3cwcmQ=",
  "port": 3128
}
```

**Note:** The passwords specified in the proxy server configuration must be base-64 encoded by using the encryptpassword.sh script.
Auditing patterns

IBM Cloud Orchestrator provides a comprehensive auditing function to help you maintain the security and integrity of your environment. This topic introduces you to the auditing capabilities, the business value of auditing, and procedures for working with the event log.

Capabilities overview

The auditing function is essentially a continuous logging activity; IBM Cloud Orchestrator records information about administrative and security-related events that occur in the product and in the cloud.

The following list displays a few examples of the events that are tracked by the auditing function:

- configuration changes
- user authentication
- attempts to access objects that are secured by object-level access control
- digital signature validation
- and more

For each event, the collected information identifies the user who initiated the operation, and whether it succeeded. IBM Cloud Orchestrator makes this audit data available for download in the form of event records.

See “Audit events” on page 275 for a complete list of tracked events.

Business value

With these capabilities you can protect your environment from both internal and external security threats. Use the audit data to identify suspicious user activity, and then hold those users accountable for their actions. In the case of an attempted security attack, analysis of the audit data can help you determine if and how your infrastructure was compromised. Based on that information, you can strategize the most effective defensive measures.

The auditing function also helps your organization to comply with regulatory laws such as the Health Insurance Portability and Accountability ACT (HIPAA) and the Sarbanes-Oxley (SOX) Act. These laws mandate formal practices not only for protecting data and detecting fraud, but also for documenting your efforts to do so. The audit data provides that evidence; with IBM Cloud Orchestrator you have numerous options for downloading the data in a manner that suits your business processes.

For detailed information on how to exploit the business value of audit event records, see “Audit event record attributes and usage tips” on page 276.

Working with the event log

The IBM Cloud Orchestrator event log stores the event records that contain audit data. To download the records, you can use any direct calls to the REST API. Note that for working with this API, IBM Cloud Orchestrator provides sample scripts that you can customize and run with a job scheduler to automate download of audit data on a regular basis. See “Download options for audit event records” on page 279 for more information.
After you download records from the log and store them in your own archives, you must delete those same records from the log. Otherwise, when the log reaches a pre-set capacity limit, IBM Cloud Orchestrator suspends the auditing function until storage frees up. When consumption nears 90%, clean the audit log storage. See "Deleting audit data to free storage" on page 284 for more information.

**Best practice:** Designate one individual with admin role to download audit data, archive it to external storage, and then delete it from the Workload Deployer component machine as a routine process.

**Audit events**

With the IBM Cloud Orchestrator auditing function, you collect data about specific types of user activity, security events, and configuration changes on the product and in the cloud. That audit data can help you detect and analyze potential security breaches, or other misuse of cloud resources. This reference article lists the events about which the product generates audit data.

Table 1 organizes the audit events in two broad categories, as indicated by the column headings.

<table>
<thead>
<tr>
<th>User activity on the product</th>
<th>User activity on cloud objects and data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every occurrence of product startup and shutdown per user</td>
<td>Security violation</td>
</tr>
<tr>
<td>Success or failure of login attempts</td>
<td>Data deletion</td>
</tr>
<tr>
<td>Every user permission assignment</td>
<td>Success or failure of attempts to access a security file or protected resource (To provide context about these attempts, IBM Cloud Orchestrator includes a list of protected resources and users who can access them.)</td>
</tr>
<tr>
<td>Configuration changes</td>
<td>Process invocation</td>
</tr>
<tr>
<td>Session timeout</td>
<td>Session timeout</td>
</tr>
<tr>
<td>Backup profile (creation, deletion or update)</td>
<td>Virtual system instance (creation, deletion or update)</td>
</tr>
<tr>
<td>Virtual machine (creation, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>Task (addition, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>Hypervisor (addition, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>IP (creation, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>IP group (creation, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>Storage device (addition, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>Network interfaces (addition, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>Pattern (creation, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>Virtual image (creation, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>Disk image (addition, deletion or update)</td>
<td></td>
</tr>
<tr>
<td>Script package (creation, deletion or update)</td>
<td></td>
</tr>
</tbody>
</table>
Event records

IBM Cloud Orchestrator collects audit data about each of the previously listed events in *event records*; one record corresponds with each event. For descriptions of event record attributes and an understanding of how to analyze the attribute values, consult the article "Audit event record attributes and usage tips."

Audit event record attributes and usage tips

Use this article as a reference aid for exploiting the business value of your audit data. The following sections define the record attributes, show examples of attribute values, and provide guidelines for using the attributes to analyze questionable or malicious user behavior in your IBM Cloud Orchestrator environment.

Record structure overview

The product captures data about each audit event in an *event record*, in comma-separated value (CSV) format. The first eight comma-separated elements of every record are values for the common attributes that are listed in Table 1 of the next section. Within each record, the values for these common attributes are followed by attribute name-value pairs that can vary from record to record. Table 2 in the section "Attribute name-value pairs" lists the pairs that you can use in your analysis of cloud activity.

**Example:** The following example depicts a short, but still typical, event record. The first line (from 3.1.0.0 to 127.0.0.1) is comprised of values for the eight attributes that are common to all records. The subsequent lines consist of attribute name-value pairs.

```
3.1.0.0,2011-11-11 02:43:11.011 UTC,user,login,1,Administrator,{system},127.0.0.1,
```

---

**Note:** In the event records you might see `cbadmin` as user ID because this is a built-in internal user of the product.

**The eight common attributes**

Table 1 provides the attribute definitions that apply to the first eight values in every event record. All of these attribute values appear in the same order in every record, and all of them are strings.

<table>
<thead>
<tr>
<th>Order of appearance in record</th>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Product version</td>
<td>Version of the IBM Cloud Orchestrator product</td>
</tr>
<tr>
<td>2</td>
<td>Timestamp</td>
<td>Time (in UTC time zone) when the event record was generated</td>
</tr>
<tr>
<td>3</td>
<td>Resource type</td>
<td>Type of resource on which an action was attempted</td>
</tr>
<tr>
<td>4</td>
<td>Action</td>
<td>Action that was attempted on the specified resource</td>
</tr>
<tr>
<td>5</td>
<td>Resource ID</td>
<td>Resource instance number</td>
</tr>
<tr>
<td>6</td>
<td>Resource name</td>
<td>Name of the specified resource instance</td>
</tr>
<tr>
<td>7</td>
<td>User</td>
<td>Who attempted to perform the specified action</td>
</tr>
<tr>
<td>8</td>
<td>Client IP address</td>
<td>IP address from which the action was initiated</td>
</tr>
</tbody>
</table>
Attribute name-value pairs

Table 2 depicts attributes that are not common to all event records because the data is not applicable to every audit event.

**Note:** You might see event records with attribute name-value pairs that are not listed in this table. Do not include those unlisted attributes in your analysis of audit data. They are subject to change in future releases of IBM Cloud Orchestrator.

Table 36. Attribute name-value pairs

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Data value example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_authz_acl_check</td>
<td>/admin/users/u-0/userdata.json_RWF_true</td>
<td>Indicates that a user with an appropriate permission (read, write, or full access permission) successfully accessed the specified resource, which in the data value example is /admin/users/u-0/userdata.json. When a user accesses multiple resources within an audit event, the product concatenates those resource names in the attribute value. For example: /admin/users/u-0/userdata.json_RWF_true_/admin/plugins/webservice/1.0.0.3/parts/webservice.scripts.tgz_WF_true</td>
</tr>
<tr>
<td>event_authz_check</td>
<td>success, failure, or reject</td>
<td>Shows the result of a REST interface access control check.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When performing this verification step, the product verifies every access request against a set of rules. These rules entail verifying the endorsement signature, the freshness of the request timestamp, the integrity of the security token, and the sufficiency of the caller and asserted security roles.</td>
</tr>
</tbody>
</table>
| event_authz_header    | [{"attributes": "\"authorizationAttributes\": \"\"
|                       |                                                                                 | Displays a stack of one or more security tokens that represent requester security credentials.                                                                                                                |
|                       | [{"attributes": "\"authorizationAttributes\": "\"
|                       |                                                                                 | Overview of the stack structure:                                                                                                                                                                           |
|                       | }]                                                                                | • The first token of a stack is known as the caller security token, or caller token, and represents the security credentials of the requesting user. (The caller token is the only token in a single-token stack.) |
|                       |                                                                                  | • Each subsequent token represents an intermediate server that relayed the request on behalf of the original caller. These tokens are endorsement security tokens, or endorsement tokens. |
|                       |                                                                                  | • Altogether, the caller token and endorsement tokens of a multi-token stack represent the path that a request travelled to reach a resource.                                                             |
|                       |                                                                                  | **Note:** The last security token in the stack represents the most recent endorsement server, which can assert additional security roles, if necessary, when making downstream request invocations. |
| event_exception       | CWZSE0924W: User: user1 not found                                                | Specifies the error condition that has caused a request failure or rejection.                                                                                                                              |
Table 36. Attribute name-value pairs (continued)

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Data value example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_outcome</td>
<td>success, failure, or reject</td>
<td>Indicates whether the overall request process was successful, a failure, or rejected.</td>
</tr>
<tr>
<td>event_request_remote</td>
<td>172.16.65.4_172.16.65.4_52917</td>
<td>Displays the requester host name, IP address, and port number.</td>
</tr>
<tr>
<td>event_request_url</td>
<td>https://workload_deployer:9444/sts/admin/registries</td>
<td>Specifies the request URL.</td>
</tr>
<tr>
<td>event_roles</td>
<td>[REPORT_READER]<em>[AUDIT.Reader]</em>[AUDIT.WRITER]</td>
<td>Lists the security roles of the specified user. The list consists of security roles that have been granted to the user, as well as any additional security role that is asserted by the most recent endorsement server. Refer to the description of the event_authz_header attribute for more information about endorsement servers.</td>
</tr>
<tr>
<td>event_subjects</td>
<td>[user1]</td>
<td>Displays the security identity of the requester. In cases where an intermediate server makes downstream requests on behalf of a user, the event_subjects attribute tracks the chain of requester identities. For example, the value [user1]_[cbadmin] indicates that an intermediate server assumed the cbadmin security identity to make a request on behalf of the original requester user1.</td>
</tr>
</tbody>
</table>

Security token format

As mentioned previously, the event_authz_header attribute displays IBM Cloud Orchestrator security tokens as signed JSON objects. Review the following example for an understanding of the data that these security tokens contain. (Note that the security token format is subject to change in future releases of the product.)

```
{
   "attributes":
   "{"authorizationAttributes" : { "groups" : ["g-0"],
   "roles" : [
   ["11","13","14","15","16","17","18","2","3","4","5","6","7","8","9","10"] },
   "ownerProcessTypeID" : "IT",
   "ownerPublicKey" : "IT",
   "AT" : "1316453534588",
   "userName" : "cbadmin",
   "userID" : "u-0",
   "type" : "user",
   "issuerProcessTypeID" : "TS",
   "expirationTime" : 86400000,
   "issuerPublicKey" : "TS"
   },
   "signature" : "IPf***A=="
}
```

Guidelines for analysis

Ultimately, the business value of audit data analysis is to minimize risk to your business assets, by maintaining the integrity of your IT practices and building effective security measures for your environment. The following guidelines and analysis scenarios give you insight to achieve those critical goals.

- Detect fraudulent or risky user activity, and take action to preserve system integrity -
Use event record attributes to track the activity of both human and non-human user entities. (Remember that a user entity might not be a human, but rather a system such as a deployed virtual machine.) For example, you might want to track the recent activity of a specific user on a specific resource. You can search event records with attributes that meet all of the following conditions:

- A User value that matches a specific user security identity
- Values for Resource type, Resource Name, and, optionally, Resource Id that match your resource of interest
- Timestamp values that correspond with your time frame of interest

If you examine the records and find user behavior that potentially compromises your environment, you can hold the offender accountable. If the user in question is actually a non-human user entity, you can make configuration modifications to minimize risk.

**Analyze security attacks to provide insight for proactive security measures -**

Use attributes to perform detailed intrusion detection and forensic analysis if an attack occurs. The attributes `event_subjects`, `event_authz_header`, and `event_authz_acl_check` are particularly helpful for these purposes. The following list enumerates the ways in which you can use the attributes:

- The `event_subjects` attribute shows the complete path of a user's request to access a resource; use it for a quick analysis of how a malicious user might have launched his or her attack.
- You can also use `event_subjects` to determine which records require detailed examination. Consider the attribute as a concise summary of the security token stack information in the `event_authz_header` attribute. Therefore, you can use `event_subjects` to pinpoint the event records that need to be analyzed meticulously with the information in `event_authz_header`.
- You can use the `Resource Type` and `Resource Name` attributes to identify records that document activity on a resource of particular concern. Then you can examine the `event_authz_acl_check` attribute for more detailed information about the user who accessed that resource. Consider the following sample `event_authz_acl_check` value:

```
/admin/plugins/webservice/1.0.0.3/parts/webservice.scripts.tgz_WF_true
```

This value indicates that the user who accessed the resource `/admin/plugins/webservice/1.0.0.3/parts/webservice.scripts.tgz` has write and full permissions for that resource. Thus, when the integrity of a resource is compromised, you can refine your list of suspected perpetrators to users who have write and full permissions for the resource in question.

Based on these tips and illustrations, you can strategize other ways to use the rich audit data that IBM Cloud Orchestrator provides to protect your environment and, consequently, your business data.

**Download options for audit event records**

You can download audit event records by using the REST API. The records that you request are provided in comma-separated value (CSV) format (wherein the attribute values of each record are separated by commas).

**File names**

You can specify the name of the .zip file to which your records are written for the download operation.
Additional files and content

Table 37. Download options for audit data

<table>
<thead>
<tr>
<th>Download option</th>
<th>Files that you receive</th>
<th>Link to download instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST API</td>
<td>XXX.zip, where XXX is a name that you specify by parameter. This .zip file contains:</td>
<td>&quot;Retrieving audit data with the REST API&quot;</td>
</tr>
<tr>
<td></td>
<td>- audit-events.csv - This file contains the audit event records that you specified, in CSV format.</td>
<td>Note: For working with the REST API, IBM Cloud Orchestrator provides sample scripts that you can customize and run with a job scheduler to automate downloads on a regular basis.</td>
</tr>
<tr>
<td></td>
<td>- audit-events-signed-events-checksum - Contains the digital signature that verifies both the integrity and authenticity of your audit data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- audit-events-record-IDs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- audit-events-signed-record-IDs</td>
<td></td>
</tr>
</tbody>
</table>

Retrieving audit data with the REST API

For retrieving audit event records with the REST API, IBM Cloud Orchestrator provides sample scripts that you can customize and run with a job scheduler to automate downloads on a regular basis. The REST API returns the event records in comma-separated value (CSV) format, in a .zip file.

Before you begin

Working with the auditing resources and scripts entails the following prerequisites:

- You need an environment that supports shell scripts, Python, Java™, Curl, and a file archiving utility (for manipulating .zip files). For example: Cygwin, a compatibility layer, meets all of these requirements.
- All of the scripts are located in CLI directories.
- You must have the admin role in IBM Cloud Orchestrator.

You also might want to review general information about the REST API in the article "REST API reference" on page 655. For this particular procedure, you need to understand that invocation of the API is accomplished through HTTP requests.

Tip: If you want to create your own scripts for retrieving audit data, consult the reference article Auditing REST API for the syntax to use in your HTTP requests. You can browse this procedure and the sample scripts for an understanding of how your code must integrate the fundamental tasks (authenticating with the REST API, issuing the HTTP requests, and specifying a .zip file for the returned data).

About this task

Table 1 lists the sample scripts that you can use to download the .zip file of audit data and unzip it. The following procedure steps provide guidelines for using the scripts.

Table 38. Overview of sample scripts

<table>
<thead>
<tr>
<th>File name</th>
<th>Functional description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_basicauth_header.py</td>
<td>This Python script is used by the following Curl script to create an HTTP basic authentication header with your user key information (your IBM Cloud Orchestrator credentials).</td>
<td>&quot;deployer,cli-XXX\deployer,cli\IBM,XXX\deployer&quot;, where XXX is the version number of the CLI</td>
</tr>
</tbody>
</table>
### Table 38. Overview of sample scripts (continued)

| File name   | Functional description                                                                                                                                                                                                                                                                                                                                                     | Location                                                                 |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cscurl.sh   | This shell script performs the following tasks:<br>• Runs the `create_basicauth_header.py` script to create the authentication header.<br>• Issues a Curl command to IBM Cloud Orchestrator with the authentication header to authenticate your credentials. The Curl command targets the URL of the IBM Cloud Orchestrator component that provides the audit-related REST API.<br>• Sends the HTTP request for downloading the audit data to the REST API. **Note:** You must specify the HTTP request as a parameter of `cscurl.sh`.                                                                                           | `...\deployer.cil-XXX\deployer.cil\lib\XXX\deployer`, where XXX is the version number of the CLI                                      |
| auditFetch.sh | This sample script provides a very simple demonstration of how you can automate the entire process, and thus adopt it as another job that you run regularly in your IT organization.                                                                                           | `...\deployer.cil-XXX\deployer.cil\samples`, where XXX is the version number of the CLI                                             |

Steps 1 - 5 are preparatory steps for using these scripts. Steps 6 - 7 describe how to use `cscurl.sh` to download audit data in a `.zip` file, and then unzip that file. Step 8 describes how to use `auditFetch.sh`. The `auditFetch.sh` script invokes both of the other scripts to automate the download and unzip operations. Consider it as a model for code that you can run with a job scheduler to regularly download audit data.

**Procedure**

1. Place the scripts in an appropriate working directory.
2. Download your IBM Cloud Orchestrator user keys from the product and save them in the same directory that contains the scripts.<br>   a. Open a browser window and go to `https://Workload_Deployer_server/resources/userKeys/`.  
   b. Type your user name and password in the authentication fields.  
   c. Select the directory and provide a file name for storing your keys.  
   d. Save the keys as a `.json` file.
3. **Optional:** For added security in a production environment, you can use the IBM Cloud Orchestrator root certificate to authenticate the scripts to the REST API. Follow these steps to download the certificate and save it in the same directory that contains the scripts and your user keys:<br>   a. Open a browser window and go to `https://Workload_Deployer_server/resources/rootcacertificate/`.  
   b. Select the directory and provide a file name for storing the root certificate. **Note:** The default file name is `cert.pem`.  
   c. Save the root certificate.
4. **Optional** (but necessary if you want to use the root certificate for authentication): In the `/etc/hosts` file of your workstation, bind the IP address of your IBM Cloud Orchestrator to the name that is used in the root certificate, `IBMWorkloadDeployer`.  
5. Construct the URL for the download request that the `cscurl.sh` script sends to the REST API; you must provide this URL as a parameter to run `cscurl.sh` in the next step.

You can choose between two options for downloading your audit data. The more basic option is to specify a maximum number of records to download. Alternatively, you can specify both a maximum number of records and the time frame in which the product logged those records. For either option, the URL
must include the location of the REST API code that downloads the data and the resource name of the option that you choose. Use the following models for your URL:

- To simply specify a maximum number of records in the request, construct a URL for the events resource and use the size parameter:
  
  "https://Workload_Deployer:9444/audit/archiver/events?size=X"

For the X variable, substitute the number of records that you want to download. You can request up to 20,000 records. If you specify a greater number, the product automatically resets your request to 20,000 records, and ultimately writes that number of records to the .zip file.

- To add a time frame to your request, construct a URL for the filteredEvents resource and specify the start and end times as Epoch timestamps. (Use the time conversion tool of your choice to convert your desired date and times into Epoch timestamps.)
  
  &startTime=long_value&endTime=long_value"

Note: If you did not perform Steps 3 - 4, replace the name Workload_Deployer in the URL with the IP address of your IBM Cloud Orchestrator server.

6. Run cscurl.sh with parameters that specify the URL that you created in Step 5, as well as the name of the .zip file in which you want the REST API to store the audit data. For example:

```
./cscurl.sh username=user1 password=user1
  keyfile=userkeys.user1.json -v --cacert root_cert.pem
  -H "Accept:application/octet-stream"
  "https://IBMWorkloadDeployer:9444/audit/archiver/events?size=X"
```

Note all of the variables that represent parameter values in the statement:

- **user1** = the user name
- **user1** = the user password
- **userkeys.user1.json** = the file that contains the user keys
- **root_cert.pem** = the name of the file that contains the IBM Cloud Orchestrator root certificate
- **X** = the number of records to be downloaded
- **ArchiveFetchTempFile** = the name of the .zip file to which the audit data is written

Also, be aware of the following usage notes for running cscurl.sh:

- To use the script to send a request for the filteredEvents resource (to retrieve event records that the product logged within a specific time frame), encapsulate the URL in single quotes rather than double quotes.
- Running cscurl.sh without parameters triggers display of its help message.
- To guard against data loss, you must specify a file for the audit data. Otherwise, the script returns it as simple command-line output.

7. Unzip the archive file that is returned from the REST API. (In response to the previous example of running the script, the REST API would return ArchiveFetchTempFile.zip.) Consequently you now have four files, as depicted in the following list:

- **audit-events.csv** - Contains your audit event records in CSV format.
At this point the retrieval process is complete. If you followed Steps 1 - 7, you successfully used the individual scripts and the REST API to write your audit data to a .zip file and download it. Step 8 describes auditFetch.sh, which automates the entire process; the script provides an example of code that you can run with a job scheduler to regularly download audit data.

8. To run auditFetch.sh, use the following statement as a model:

```
./auditFetch.sh username=auditor password=auditor
keyfile=userkeys.auditor.json IWD=IP address size=X > ArchiveFetchTempFile
```

**Tip:** If you performed Steps 3 - 4 in order to use the root certificate to authenticate with the REST API, modify the script's invocation of cscurl.sh as follows: Replace the -k option with the -v --cacert options. Remember to specify the name of the file in which the IBM Cloud Orchestrator root certificate is stored on your workstation, as in: -v --cacert cert.pem. Then when you type the statement to run auditFetch.sh, change the value of the IWD parameter to the name IBMWorkloadDeployer.

Based on the values that you supply for the parameters, auditFetch.sh performs these tasks to download your audit event records:

- Authenticates your credentials with IBM Cloud Orchestrator.
- Constructs an HTTP request for X number of records and sends it to the REST API. (If you do not specify a particular number of records with the size parameter, the REST API returns a maximum number of 20,000 records.)
- Stores the records that are retrieved by the REST API in a .zip file with the name that you specified. (If you do not specify a file name, the script uses the default name of ArchiveFetchTempFile.zip.)
- Unzips that archive file, which contains the following files:
  - `audit-events.csv` - Contains your audit event records in CSV format.
  - `audit-events-signed-events-checksum` - Contains the digital signature that verifies both the integrity and authenticity of your audit data. Archive this file along with your event records.
  - `audit-events-record-IDs`
  - `audit-events-signed-record-IDs`

**Note:** The last two files contain data that you must send back to the REST API to delete the event records from the Workload Deployer machine (to free storage resources). See the "What to do next" section of this article for more information about deleting audit data.

Be aware of the following usage notes for running auditFetch.sh:

- Running this script without parameters triggers display of its help message.
- In some environments, such as Ubuntu, you might need to modify the script's invocation of cscurl.sh; add the period and forward-slash characters (./) as a prefix to the statement that invokes cscurl.sh.
What to do next

Because IBM Cloud Orchestrator does not automatically delete audit data after you download it, you must run the auditDelete.sh script to delete the data from the Workload Deployer machine and free storage resources. You can use this script along with your customization of auditFetch.sh as part of a regularly scheduled job to download, archive, and then delete audit data. See the article “Deleting audit data to free storage” for information about auditDelete.sh.

Deleting audit data to free storage

To help safeguard your auditing processes, IBM Cloud Orchestrator does not automatically delete audit event records from the event log after you download them. Instead, you must delete the records from the log in order to free storage on the Workload Deployer machine. This article describes how to delete event records with the auditDelete.sh script, which exploits the IBM Cloud Orchestrator REST API.

Before you begin

- Understand that when audit data storage reaches a pre-set capacity limit on the machine where the Workload Deployer component has been installed, IBM Cloud Orchestrator suspends collection of audit data until the storage resources free up. Therefore, it is recommended that you delete audit data from the event log immediately after you download the data and archive it, preferably in a routine job. If you choose not to adopt this best practice, always remove the oldest record sets first when you clean the audit log.

- Also consider these other best practices for deleting audit data from the Workload Deployer machine:
  - Designate one individual with admin role to download audit data and subsequently delete it from the Workload Deployer machine.
  - Do not run download and delete operations concurrently.

- You need an environment that supports shell scripts, Python, Java, and Curl. For example: Cygwin, a compatibility layer, meets all of these requirements.

- The auditDelete.sh script is located in the deployer.cli samples library.

- Finally, you must have the admin role to delete audit event records.

About this task

When you run auditDelete.sh, you clean up specific record sets that you have already downloaded from the Workload Deployer machine; you identify these record sets with the parameters that you specify for the script.

Note: the auditDelete.sh script needs cscurl.sh and cscurl.sh requires create_basicauth_header.py to run. Therefore, these files (cscurl.sh and create_basicauth_header.py) must be in the same folder as the script. If you want to use this sample script, you must manually copy the required files from the library.

Procedure

1. Locate the auditDelete.sh script in the samples library of the IBM Cloud Orchestrator command-line interface (CLI). The directory path is ...\deployer.cli-XXX\deployer.cli\samples, where XXX is the version number of the CLI. Copy the script to another directory if you wish.
2. Download your IBM Cloud Orchestrator user keys from the Workload Deployer machine and save them in the same directory that contains the auditDelete.sh script.
   a. Open a browser window and go to https://<your_workload_deployer_server>/resources/userKeys/.
   b. Type your user name and password in the authentication fields.
   c. Select a directory and provide a file name for storing your keys.
   d. Save the keys as a .json file.
3. To run the script, use the following statement as a model:
   ```
   ./auditDelete.sh username=auditor password=auditorpswd
   keyfile=userkeys.auditor.json IWD=<your_workload_deployer_server>
   map=record_IDs hash=signed_hash
   ```
   Note all of the variables that represent parameter values in the statement:
   * `auditor` = Your user name
   * `auditorpswd` = Your user password
   * `userkeys.auditor.json` = The file that contains your user keys
   * `your_workload_deployer_server` = IP address of the machine where the Workload Deployer component has been installed.
   * `record_IDs` and `signed_hash` - Both variables, which represent values for the parameters `map` and `hash`, identify the record set to be deleted from the Workload Deployer machine. These parameter values name two files that were products of a previous download operation, and were included in your audit-events.zip file. Define the parameters as follows:
     - For the `map` parameter, specify the name of the file that lists the IDs of the records that you downloaded. It was included in your downloaded .zip file as audit-events-record-IDs.
     - For the `hash` parameter, specify the name of the file that contains the signed hash of the record IDs. It was included in your downloaded .zip file as audit-events-signed-record-IDs.
   However, you do not need to specify the `map` and `hash` parameters if all of the following conditions are true:
     - You are deleting records that you just downloaded from the Workload Deployer machine.
     - You have not changed the name of either audit-events-record-IDs or audit-events-signed-record-IDs.
     - You have placed both files in the same directory as auditDelete.sh.

Results

You have now deleted the previously downloaded audit event records from the Workload Deployer machine.

What to do next

Review the article “Audit event record attributes and usage tips” on page 276 for an understanding of how you can best exploit IBM Cloud Orchestrator event records in your auditing analyses.
Password management

There are two ways of managing your password in IBM Cloud Orchestrator.

Changing password

You can change your password from the Administration user interface.

Procedure

1. Click Settings in the drop down menu under the logged in user name in the Administration user interface. The Settings panel appears.
2. Select the Change Password tab.
3. In the Change Password panel enter the current password, followed by the new password, and click Confirm.

Results

The password has been changed successfully.

Resetting a password

You can reset a forgotten password using the keystone command line interface tool.

Procedure

1. As root log onto Central Server 1.
2. Source the keystone rc file by running the following command: source keystone rc.
3. Use the keystone user-get command to see the unique id related to the user whose password you want to change.
4. Once you have the id, use the keystone user-password-update command to reset the password. The format of the command is as follows: keystone user-password-update [--pass <password>] <userId>. The password parameter is the password you want to update to. The userId is the unique user Id retrieved from the user-get command.

Results

The password has been reset.
Chapter 4. Managing orchestration workflows

Create custom orchestration workflows in the Business Process Manager user interface and run them in your IBM Cloud Orchestrator environment.

Orchestration workflows

You can use three types of orchestration workflows:

- "Managing offerings" on page 319: These workflows are used to define the offerings that cloud users can select in the Self-Service Catalog. They include user interface and the service request flow.
- "Managing actions" on page 322: These workflows are used to define additional actions that cloud users can run on a given running instance. You can initiate these actions with the More Actions menu available in Instance view in the IBM Cloud Orchestrator user interface.
- "Orchestration actions for Virtual System (Classic) Patterns" on page 288: Orchestration actions are provided for backward-compatibility with IBM SmartCloud Orchestrator v2.3. These workflows are used to define additional Orchestration actions for Virtual Systems (classic). Orchestration Actions are user actions or event-triggered actions. You can initiate user actions with the More Actions menu available in the instance view for virtual systems (classic). Event-triggered actions can be called during the deployment cycle of virtual systems (classic) at various points so that the deployment process can be customized, enhanced or overridden in some way.

An orchestration workflow, which is based on Business Process Manager Business Process Definition, defines a logical flow of activities or tasks from a Start event to an End event to accomplish a specific service. The service can be started either by events triggered by IBM Cloud Orchestrator management actions or by user actions in the IBM Cloud Orchestrator user interface. The activities that comprise the service can be either scripts (JavaScript), Java implementations, web services or REST calls, human tasks, and so on. They can be either executed in sequence or in parallel with interleaving decision points.

Each activity within an orchestration workflow has access to the cloud environment data in the form of the OperationContext object, which is passed as input parameter to each orchestration workflow. The operation context is an umbrella object that contains all data that is related to the execution of an operation. The operation context object must be defined as an input parameter variable for all business processes that are started as an extension for an IBM Cloud Orchestrator operation. Human services must define the operation context ID as an input parameter and as a first activity, must retrieve the operation context object with its ID. The operation context object contains metadata information, for example:

- User
- Project
- Event topic
- Status

It also contains information about the instance in which the orchestration workflow is executed, for example:
• Type
• Status
• Virtual system ID
• Virtual system pattern ID
• Virtual application ID
• Information about the virtual machines that belong to the instance - CPU, memory, disk space, and so on.

For more technical details about the operation context object, see the IBM Cloud Orchestrator Content Development Guide.

Workflows can throw error events or post-status messages, which are then shown in the IBM Cloud Orchestrator user interface. For more information about errors, see the IBM Cloud Orchestrator Content Development Guide.

An orchestration workflow can also have additional user interface panels in order to collect data that is needed as input. These panels are also implemented based on workflow technology, and they are called human services in Business Process Manager.

Self-service offerings

Self-service offerings are typical administrative actions that are used to automate the configuration process.

Offerings, like actions, are custom extensions to IBM Cloud Orchestrator. You can develop these extensions by using Business Process Manager Process Designer, and then add them as offerings in the CONFIGURATION tab in the IBM Cloud Orchestrator Self-service user interface. An offering can consist of:
• A Business Process Manager business process defining the activities to be performed by the extension.
• User interface panels that collect additional data, implemented by a Business Process Manager human service (optional).

Users access offerings in the SELF-SERVICE CATALOG tab, where they are grouped into categories.

Related tasks:
“Managing self-service offerings” on page 318
A Service Designer can create self-service offerings and use them to customize the IBM Cloud Orchestrator environment. A Service Designer is a user with the catalogeditor role.

Orchestration actions for Virtual System (Classic) Patterns

Orchestration actions refer to actions that are run on virtual system instances only.

Click PATTERNS > Pattern Design > Orchestration Actions to create, modify or delete Orchestration Actions. There are two types of Orchestration Actions, User actions and Event-triggered actions.
User actions
User actions are custom actions that can be run on virtual system instances.

User actions implement additional lifecycle management actions which extend the set of predefined actions.

To view the list of all available user actions click PATTERNS > Pattern Design > Orchestration Actions. Within this view you can also add new actions and modify or delete existing actions. To search for an action enter the action name or description in the search field.

Event-triggered actions
Event-triggered actions are Business Process Manager business processes that are triggered by a specified event during a predefined management action for a classic virtual system.

To view the list of all available event-triggered actions, click PATTERNS > Pattern Design > Orchestration Actions. Within this view you can also add new actions and modify or delete existing actions. To search for an action enter the action name or description in the search field. You can customize event-triggered actions to run during events that are called plug points. The plug points are categorized based on actions during which they occur:

- Deployment or undeployment of a pattern:
  - Before provisioning
  - After provisioning
  - Before start of virtual system instance
  - Before virtual system instance deletion
  - After virtual system instance deletion

- Server actions, like Start, Stop, Delete, and Modify Server Resources:
  - Before the instance status changes to start
  - Before the instance status changes to stop
  - After the instance status changes to start
  - After the instance status changes to stop
  - Before the server status changes to start
  - Before the server status changes to stop
  - After the server status changes to start
  - After the server status changes to stop
  - Before the server status changes to delete
  - After the server status changes to delete

Note: A user interface must not be defined if the related pattern is handled via a self-service offering
Samples and standard extensions for orchestration workflows

IBM Cloud Orchestrator provides an easy way to build the most common types of workflows. No programming is required. With IBM Process Designer, a graphical tool to build workflows, and the pre-built samples and templates, you can create your own workflows with simple drag and drop editing, and attach common types of automation routines that are widely available.

Pre-built samples

IBM Cloud Orchestrator includes a set of toolkits that contain templates that you can reuse and adapt to your needs.

The SC0rchosstrator_toolkit provides the essential building blocks, which are needed to build Business Process Manager business processes and human tasks, which are then used as extensions for IBM Cloud Orchestrator.

Note: This toolkit and all the other provided toolkits can be found in [IBM Cloud Orchestrator content development](https://www-304.ibm.com/software/brandcatalog/ismlibrary/cloudorchestratorcatalog).

You can also search for additional samples in the IBM Cloud Orchestrator Catalog at [https://www-304.ibm.com/software/brandcatalog/ismlibrary/cloudorchestratorcatalog](https://www-304.ibm.com/software/brandcatalog/ismlibrary/cloudorchestratorcatalog). IBM Cloud Orchestrator Catalog is a platform and one-stop-shop for IBM customers, partners, and employees, where developers, partners, and IBM Service teams continuously share content among one another.

Advanced programming

To create more sophisticated automation, involving richer programming languages, see [IBM Cloud Orchestrator content development](https://www-304.ibm.com/software/brandcatalog/ismlibrary/cloudorchestratorcatalog).

Working with Business Process Manager

You can use Business Process Manager processes to extend the capabilities of IBM Cloud Orchestrator.

About this task

Business Process Manager provides two user interfaces with which you can work to extend the capabilities of IBM Cloud Orchestrator - Process Center and Process Designer. You can switch between them to use different functions of the product.

Process Center is a web-based application that you launch from the IBM Cloud Orchestrator user interface. In this application, you can review and manage process applications and toolkits that are known to the process server.

Process Designer is a stand-alone application that you install manually on a local computer. It includes a Process Center view that provides access to the repository but is enhanced with an Open in Designer option, with which you can design and configure your own workflows. You use the Process Designer graphical tool to create custom extensions to IBM Cloud Orchestrator. Pre-built sample artifacts are provided in Business Process Manager to make process creation quick and easy.

For detailed information on IBM Business Process Manager, refer to the [information center at](http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r5m0/topic/com.ibm.wbpm.main.doc/ic-homepage-bpm.html).
### Setting up IBM Process Designer

Install, configure, and log in to IBM Process Designer.

IBM Process Designer is a stand-alone, local application that you install on a Windows operating system computer.

1. In a web browser, log on to the Business Process Manager user interface as user `admin` and with the password set for `admin` during installation.

2. Install Process Designer on a Windows machine on which you design the workflows:
   
   a. On the right-side panel of Process Center, click **Download Process Designer**. This is a link to the Process Designer installation package.
   
   b. Install the package as described in **Installing IBM Process Designer** in the Business Process Manager information center.

3. Click **Start > IBM Process Designer Edition > Process Designer** and log on as user `admin` with password `passw0rd`.

Process Designer stand-alone application opens and a list of process applications is displayed in the **Process Apps** tab. When you click the process application name, you can view its details, such as snapshots, history, you can edit some details such as name, or who can access it, but you are not able to configure the process application in this view. To configure a process application, click **Open in Designer** next to the item name.

You can switch between Designer, Inspector, and Optimizer tabs.

- To plan and design processes, use the Designer view.
- To test and debug processes, use the Inspector view.
- To analyze and optimize processes, use the Optimizer view.

To return to Process Center view, click **Process Center** in the upper right corner of the screen. In the Process Center view, click **Open in Designer** to get back to the Designer view.

### Adding users to IBM Process Designer

If you want a new IBM Cloud Orchestrator user to be able to use Process Designer, you must grant it access to the Process Center repository.

2. Go to **Admin > Manage Users**.
3. Click **Add Users/Groups** on the right side of the panel. A dialog box opens.
4. In the **Search for Name** box, enter the user name.
5. Select your user in the **Results** box and click **Add selected**.

### Creating a process application in Process Designer

Create a process application and search through artifacts.

*Process applications* are containers in the Process Center repository for process artifacts such as process models and supporting implementations that are created in Process Designer.

*Toolkits* are collections of process assets that can be shared and reused across multiple projects in Process Designer.
Toolkits enable Process Designer users to share library items across process applications. Process applications can share library items from one or more toolkits, and toolkits can share library items from other toolkits.

IBM Cloud Orchestrator provides a toolkit that is called SCOrchestrator_Toolkit. It provides basic capabilities with which you can design orchestration processes. Any process application that contains processes for use with IBM Cloud Orchestrator must depend on this toolkit.

When you create a dependency on a toolkit, you can use the library items from that toolkit for the implementation of the process steps you are building in your current project. For example, after you create a dependency on a toolkit that includes several services, the Designer view automatically makes those services available when a developer is choosing the implementation for an activity.

1. Open the Process Designer and log on with administrative credentials. The Process Center panel is displayed. In this panel, you can review and manage process applications and toolkits that are known to the process server.

   **Tip:** In the Toolkits tab, notice SCOrchestrator_Toolkit.

2. Create a process application:
   a. Click the Process Apps tab and in the panel on the right side, click **Create New Process App**.
   b. In the window that is displayed, provide a name and a unique acronym for your new process application. Optionally, provide a description.

      **Remember:** Once the process application is created, do not change its acronym, as it is used to reference the processes in self-service offerings.
   c. Click **Create**.

      **Tip:** Steps 2a-c can be performed in both Process Designer and Process Center, with the same result, but only in Process Designer view, you can configure the process application when it is created.

3. Click **Open in Designer** for your newly created process application. The Designer view is opened.

4. In the Designer view, click one of the categories from the pane on the left. A list of artifacts that are relevant for this category is displayed. In this pane, you can also review the existing artifacts and add new artifacts to toolkits or process applications.

   **Note:** You can click **All** in the newly created process application to see that it initially contains only one artifact.

5. Create a dependency on SCOrchestrator_Toolkit:
   a. Make sure the process application for which you are creating a toolkit dependency is open in the Designer view.
   b. Click the plus sign next to **Toolkits** in the library.
   c. From the **Add Dependency** window, click to select **SCOrchestrator_Toolkit**.

6. Click **TOOLKITS > System Data** from the list to open the toolkit. It is one of the default toolkits. Click **All** and then choose **Tag**, **Type**, or **Name** from the drop-down list to the right to sort the artifacts by tag, type, or name.

7. Click the arrow in the upper right corner to toggle between expanding all groups or expanding only one group.
Tip: You can type any set of characters on the keyboard to search for artifacts that contain these characters.

**Reusing processes and human services in a process application**

After you create a process application, you can create processes and human services within this process application. You can also use the items that already exist in other process applications or toolkits.

In this task an example of the process application created in “Creating a process application in Process Designer” on page 291 is used.

1. In the Process Designer view search for SCOrchestrator_Toolkit.
2. On the right side of the toolkit name, click **Open in Designer**. Details for the toolkit are displayed.
3. From the list of available items on the navigation pane on the left, in the **User Interface** section, right-click the **Template_HumanService**. That is the user interface that you want to copy into your process application. The contextual menu for the selected item opens.
4. In the contextual menu, click **Copy Item to > Other Process App**. Select your process application from the list. The process is copied in the background. No confirmation is provided.
   Repeat steps 3. and 4. for all the items that you want to copy.
5. To return to the list of process applications and toolkits, click **Process Center** in the upper right corner of the screen.

When you open your process application, the **Template_HumanService** is now available on the list.

**Editing process applications and toolkits**

You must be the author of a process application or a toolkit, if you want to edit them.

### Table 39. Editing process applications and toolkits

<table>
<thead>
<tr>
<th>Action</th>
<th>Authorization required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editing imported toolkits</td>
<td>admin</td>
</tr>
<tr>
<td>Editing current Business Process Manager system toolkits</td>
<td>tw_admins, tw_authors</td>
</tr>
<tr>
<td>Creating a new process application and referencing SCOrchestrator_toolkit as a dependency</td>
<td>admin</td>
</tr>
<tr>
<td>Opening sample applications as admin</td>
<td>tw_admins group added to the Process Center (open Manage Access to Process Library)</td>
</tr>
</tbody>
</table>

**Creating a process**

Create a process using IBM Process Designer and incorporate a new activity into it.

In this example, create a sample process called "Hello World". You can modify this procedure to suit your needs.

1. In the Designer view, click the newly created process application and then click the plus sign next to **Processes** to open the **Create new** menu.
2. From the menu, select **Business Process Definition**.
3. Provide a name for the new business process definition, for example **Hello World**. Click **Finish**. The new process definition opens in the main canvas. Initially, the diagram view contains two lanes:
   - The System lane that is the default lane for system activities.
   - The Participant lane that is the default lane for user services.

   The start event and the end event are added automatically.

4. To add a user activity to the process, select **Activity** from the palette to the right. Add the activity to the Participant lane. An activity represents one step in the process. Properties of an activity are shown in the bottom panel.

5. In the Properties panel at the bottom of the screen, you can set the name of the activity, for example **Say Hello**.

6. To make the activity part of a flow, connect it with the start and end event. Select **Sequence Flow** from the palette.

7. With the **Sequence Flow** tool, click the connection points of the elements. First, connect the start event with the activity and then connect the activity with the end event.

8. To create an implementation for this process, click the plus sign next to **User Interface** in the Designer view.

9. From the menu that opens, select **Human Service** and name it **Say Hello**. The main canvas opens. You can now use the **Coach** element to create a simple user interface dialog that brings up the **Hello World** string.

   **Note:** The checkered background indicates that an implementation is being designed. It also indicates that there are now some elements in the palette that were not available in the process definition step.

10. Drag the **Coach** element onto the canvas. Specify its name, for example **Say Hello**, and double-click the element. The Coaches tab opens.

11. Drag the **Output text** element onto the canvas. In the Properties panel at the bottom of the screen, change the label to **Hello World**.

12. Drag the **Button** element and change its label to **OK**.

13. Click **Diagram** to open the Diagram tab. Use the **Sequence Flow** tool to connect the start event with the **Say Hello** step. Then, connect the **Say Hello** step with the end event.

   **Note:** Notice the **OK** label of the connection between the **Say Hello** step and the end event. It indicates that the **OK** button is used to trigger the transition to the end event.

14. Click **Save** to save the new human service.

15. From the list at the top, select **Hello World** to switch back to the previously created process definition.

16. Click the **Say Hello** step. In the Properties tab at the bottom, click **Implementation** and click **Select**.

17. From the list, choose the **Say Hello** human service that you created. Save the process definition.

   **Note:** Notice that the implementation has changed to **Say Hello**.

18. Click **Run Process** in the upper right corner. The view automatically switches to the Inspector view. Notice that there is one active **Hello World** process instance. The Diagram view at the bottom shows that the instance is at step 2 and that there is a task that is waiting to be performed.
19. Click **Run the selected task** in the upper right corner to run the task. The Pick User From Role window opens. Select administrative user to run the task.

**Note:** The selection of users in this window is based on the Participant Group setting of the Participant lane.

A browser window opens that shows the simple user interface that you have defined. Such user interface is called a coach.

20. In the user interface, click **OK** and close the browser.

21. In the Inspector view, click **Refresh**. Notice that the **Say Hello** task is closed and the **Hello World** process instance is completed.

### User input required at service request time

You can configure processes which require manual input from the user.

When manual input is required for a request to complete, the user gets a task assignment in the **INBOX** tab in IBM Cloud Orchestrator. You must claim the task by clicking **Claim**, before you can work with the task. If the task was already claimed by another user, the **Taken** status is displayed. There are two types of tasks:

- **Approval requests**
  - To approve or reject a selected request, click **Accept** or **Reject** accordingly. Optionally, you can provide a comment in the **Reason** field.

- **General tasks**
  - These are all tasks that are not of the approval type. Such tasks include a Business Process Manager human service. When you select the task from the list, the Business Process Manager coach opens. Provide any required parameters and click **Submit**.

### Making a new process available as a self-service offering

You can make a process that you created available as a self-service offering in IBM Cloud Orchestrator.

1. In Business Process Manager, expose the process to make it available in the IBM Cloud Orchestrator user interface:
   - Open IBM Process Designer.
   - Select your process and switch to the Overview tab.
   - In the Exposing section, click **Select** in the **Expose to start** row.
   - Select the **All Users** participant group or any other group that you want to expose the process to, and save the setting.

**Tip:** A similar procedure must be performed to make a user interface (human service) visible in IBM Cloud Orchestrator:
   - In Process Designer, select the human service and open the **Overview** tab.
   - In the Exposing section, click **Select** in the **Expose to** row.
   - Select the **All Users** participant group or any other group that you want to expose the process to, and save the setting.
   - In the **Expose as** row, click **Select**.
   - Select **URL** and save the setting.

2. In IBM Cloud Orchestrator, create an offering that is based on the process, and a category for it:
   - Log on to the Self-service user interface. You must be assigned the **catalogeditor** role or the **admin** role.
If no categories other than All Uncategorized Offerings exist for self-service offerings, you need to create at least one category. Offerings by default belong to All Uncategorized Offerings and they cannot be accessed in the Self-Service Catalog unless they are moved to another category.

b. Click CONFIGURATION > Categories.
c. Click the green plus icon to create a category. A new dialog opens where you can specify its details.
d. Click OK to create the category.
e. Click CONFIGURATION > Offerings.
f. Click the green plus icon to create an entry. A new dialog opens.
g. Provide a name, and a description for the offering.
h. Select an icon for the offering.
i. Select a category from the list.
j. From the Process drop-down list, select your process.
k. If the selected process requires user interaction, specify a user interface by selecting the related Human Service.
l. Click OK to create the offering.
m. To allow other users to access the new offering, select the offering and add the project to which the users belong to the Access granted to field.

The user can now access the offering in the Self-Service Catalog, and can request the offering.

**Making a process available as an orchestration action**

You can create a process in Business Process Manager and then use it as an orchestration action for virtual system patterns (classic).

There are two types of orchestration actions:

*Event-triggered actions* are business processes that are triggered by a specified event during a predefined management action. They may include a human service to gather additional parameters that are required to complete the automation. These actions are related to provisioning and deprovisioning.

*User actions* are triggered manually by a user in the Virtual System Instances (Classic) view after the pattern is deployed. User actions are, for example, resetting server password or backing up a server. They may include a human service.

1. In Business Process Manager, expose the process to make it available in the IBM Cloud Orchestrator user interface:
   a. Open IBM Process Designer.
   b. Select the process and switch to the Overview tab.
   c. In the Exposing section, click Select in the Expose to start row.
   d. Select the All Users participant group or any other group that you want to expose the process to, and save the setting.

The process is now visible in IBM Cloud Orchestrator.

**Tip:** A similar procedure must be performed to make a user interface (human service) visible in IBM Cloud Orchestrator:

a. In Process Designer, select the human service and open the Overview tab.
b. In the Exposing section, click Select in the Expose to row.
c. Select the All Users participant group or any other group that you want to expose the process to, and save the setting.

d. In the Expose as row, click Select.

e. Select URL and save the setting.

2. In IBM Cloud Orchestrator, create an orchestration action based on the process:

a. Log on to IBM Cloud Orchestrator. You must be assigned the catalogeditor role or the admin role.

b. Click CONFIGURATION > Actions Registry.

c. In the actions menu, click Create Action. A new dialog opens.

d. Provide a name and, optionally, a description for the action.

e. From the Action type list, select User.

f. Select one or more virtual system patterns to which the action is applied.

g. Select the process that you exposed in step 1.

h. If the selected process requires user interaction, specify a user interface by selecting the related Human Service.

i. Specify sequence priority if you want a specific run order to be applied. For actions that have the same event and priority defined, the order is unspecified.

j. Click Configure Access Control to create the action.

k. To allow other users to access the new action, select the action and add the project to which the users belong to the Access granted to field.

The action is now configured. If it is an event-triggered action, it is started automatically for selected virtual system patterns at the time the selected event takes place. You can now start the user action in the Virtual System Instances (Classic) view that you can open by clicking PATTERNS > Instances > Virtual System Instances (Classic). If user interaction is required for the configured action to complete, the user receives a new assignment in the INBOX tab.

Upgrading a process on a development system or production system

IBM Cloud Orchestrator enables you to distinguish between development mode and production mode. You can define different upgrade methods for a process, depending on whether IBM Cloud Orchestrator is configured as a development system or as a production system.

Development mode

When IBM Cloud Orchestrator is configured in development mode, the engine always calls the most recent version of process code.

In IBM Process Designer, the most recent version of process code is called Current or TIP. Development mode is especially convenient during the development of processes, because it is not necessary to create snapshots for each code change to be tested. Instead, the already running process instances (also known as inflight process instances) use the new modified code for the remaining part of the flow.

Development mode is configured by default after installation. For details of how to configure development mode, see “Configuring development mode” on page 298.

Production mode
When IBM Cloud Orchestrator is configured as a production system, the engine uses a snapshot version of the toolkit or process application code instead of the latest version of code.

A Business Process Manager process must be called by a dedicated snapshot id, so that any running process continues to use that code version, even if a new snapshot is imported into the production system while the process is running.

The snapshot version used in production mode is determined as follows:

**Toolkits**
A production system uses the most recent snapshot (of the toolkit) that is available at the start time of a process instance.

**Process applications**
In general, a production system uses the most recent snapshot (of the process application) that is available at the start time of a process instance. However, an administrator can specify a default version of a process application. If a default snapshot is configured, the production system uses the default snapshot instead of the most recent snapshot.

For details of how to configure production mode, see "Configuring production mode."

**Configuring development mode**
Development mode is configured by default after installation. You do not need to explicitly configure a system in development mode unless you are changing a production system to a development system.

To change IBM Cloud Orchestrator from a production system to a development system, complete the following steps:

1. Log on to the Business Process Manager WebSphere Application Server console as an administrator user:
   
   `https://$central-server-4:9043/ibm/console/logon.jsp`

   where `$central-server-4` is the IP address of Central Server 4.

2. In the console navigation tree, click **Servers > Server Types > WebSphere application servers > server_name > Process definition > Java Virtual Machine > Custom properties.**

3. Select the **ORCHESTRATOR_DEVELOPMENT_MODE** variable.

4. To specify development mode, set the value of the **ORCHESTRATOR_DEVELOPMENT_MODE** variable to true.

5. Set the description of the **ORCHESTRATOR_DEVELOPMENT_MODE** variable to **Activate Development Mode**.


IBM Cloud Orchestrator is now configured as a development system.

**Configuring production mode**
To configure production mode, IBM Cloud Orchestrator must be manually configured to use a snapshot instead of the latest version of code.

To configure IBM Cloud Orchestrator as a production system, create a property in Business Process Manager to define the operational mode, as follows:
1. Log on to the Business Process Manager WebSphere Application Server console as an administrator user:
   https://$central-server-4:9043/ibm/console/logon.jsp
   where $central-server-4 is the IP address of Central Server 4.
2. In the console navigation tree, click Servers > Server Types > WebSphere application servers > server_name > Process definition > Java Virtual Machine > Custom properties.
3. Click New to create a new variable called ORCHESTRATOR_DEVELOPMENT_MODE.
4. To specify production mode, set the value of the ORCHESTRATOR_DEVELOPMENT_MODE variable to false.
5. Set the description of the ORCHESTRATOR_DEVELOPMENT_MODE variable to Activate Production Mode.
   Optional: To specify a default snapshot for a process application, complete the following steps:
7. Activate the process application snapshot, as follows:
   a. Open the IBM Process Designer.
   b. Click the Process Apps tab.
   c. Select the process application.
   d. On the Snapshots page, expand the target snapshot, and then select Activate.
8. Specify the default process application snapshot, as follows:
   a. In a web browser, open the Process Admin Console.
   b. Click the Installed Apps tab.
   c. Select the process application.
   d. In the right pane, click Make Default Version.

IBM Cloud Orchestrator is now configured as a production system.

**Guidelines for working with Business Process Manager**

When you create toolkits or process applications, there are some best practices to be followed in naming conventions, structuring, modeling, and error handling.

**Guidelines for naming and documenting your toolkit or process application**

When you create toolkits, use the following naming conventions:

- Name the toolkit after the utility or services it provides.
- Add words like "Toolkit" or "Framework" so that you can differentiate it from other process applications.
- Avoid long names. You must use fewer than 64 characters.
- White spaces between words can be added if it improves readability.
- Avoid the version number in the name, unless you want to bring attention to the major changes in the solution.
- Add more information about the toolkit in Description field.
- Choose an acronym for your toolkit. Do not use the prefix "IC" as it is used for content delivered by IBM.
- Name your snapshots according to this scheme: AABB_YYYYMMDD. Exported TWX archives of your toolkit get this snapshot name appended, so you can easily identify the exported snapshots later.
AA  The IBM Cloud Orchestrator release that is prerequisite for the toolkit or process application, for example, 24 for IBM Cloud Orchestrator V2.4.

BB  Counting up the version of the toolkit, for example, 00 for the first release, and 01 for the second release

YYYYMMDD  
The date the snapshot was created

- When updating an existing process application or toolkit, do not change the chosen acronym because it is used to reference the processes in self-service offerings.

Guidelines for creating artifacts in a toolkit
The general best practices are as follows:

- In the documentation field for a Business Process Manager artifact, enter a description of the input and output parameters of that artifact.
- Use the note object of Business Process Manager to improve the readability of complex processes and human services.
- As mentioned in the naming conventions, provide an understandable and meaningful name for your artifacts.
- Keep the interface definition between a Business Process Manager Human Service and its associated Business Process Manager Business Process Definition as short as possible. The interface is defined by a Business Process Manager Business Object. This object is used to correlate a business process with its associated human services in the IBM Cloud Orchestrator UI. Use the Business Process Manager human service only to collect the parameters that are needed by its associated business process. Implement the business logic in the business process. It also helps if you enable the business process to be called by using the REST API from an external application, such as a portal application.
- Avoid Pre and Post execution assignments. Instead, add explicit activities, if needed. The execution assignments are hidden in the Business Process Manager Process Designer, and the logic of the corresponding activity or service becomes difficult to understand. If needed, use the Pre and Post executions to make simple assignments like initializing the associated Business Process Manager artifact. For example, consider having two consecutive coaches in a human service. In such cases, do not initialize the objects that are used by the second coach as being Post execution assignment of the first coach. If needed, do the initialization as a pre-execution assignment of the second coach.
- Do not use passwords in environment variables or other artifacts that are visible to everyone.
- When you deliver a solution for IBM Cloud Orchestrator, make sure that there are no validation errors. These errors can be seen in the Process Designer.
- Avoid changing the interface of a building block that is delivered as a part of a toolkit. If you change the interface of building blocks in a toolkit, it becomes cumbersome for all its dependent toolkits or Process Applications. Even changing the name might lead to redoing the mapping for all activities or services that use the building block.
Guidelines to structure your solution

- In general, an extension content solution for IBM Cloud Orchestrator consists of a Business Process Manager Process Application and a Business Process Manager Toolkit.

  The basic rule is that a process application contains artifacts that are ready to be used by the user and not meant to be changed or adapted to be useful. All other artifacts are better placed in a toolkit.

- When structuring your solution, always consider the visibility of your artifacts. Artifacts of one process application are not visible by default to another process application.

  For example, a Business Process Manager, process A, can be called by another Business Process Manager, process B. The ‘Linked Process’ activity is used if both are in the same process application or if process A is in a dependant toolkit.

  Avoid cyclic dependencies, that is, when toolkit A depends on toolkit B, avoid having a dependency on toolkit A. If such a cyclic dependency occurs, restructure your toolkits to resolve it.

- Use Business Process Manager tags and smart folders to structure your solution to make it more understandable. If you have UI parts that can be used in UI panels, define them as Coach views. These views can be reused in different Coaches. If you must change something later, for example, wording, you change only the reusable Coach view.

Guidelines for handling errors

A IBM developerWorks article explains extensively about exception handling and logging from a business process management perspective. See Related Links. It identifies the types of exceptions that are encountered in a Business Process Manager scenario. Also, it shows you how to handle them using IBM Business Process Manager.

The following are best practices in error handling:

- Use the PostMessage integration service that is delivered as a part of the SCOrchestrator_Toolkit to report messages that occur in Business Process Manager processes or Human Services back to the IBM Cloud Orchestrator UI. For event and instance operations, these messages are written to the history section of the pattern instance. For self-service offerings, these messages are written to the operationContext and are displayed in the UI.

- Define error message as localization resources.

- Raise errors in your integration services or processes using the Error End Event node.

- Catch raised errors raised from integration services using Intermediate Error Events or Event Subprocesses.

- For Java classes that are used in Business Process Manager processes or human services, define logging framework. For example, java.util.logging to log messages to the WebSphere log.

- Use the logging capabilities of Business Process Manager to log messages to the WebSphere log. A good practice is to log in the entry and exit of an activity to support debugging better.

Related information:
You can find many documents with guidelines and best practices about business process modeling. One of it is *Five Guidelines to Better Business Process Modeling for Execution* from Jonas A. Zahn and Stuart Jones, which describes the following design guidelines:

- **Rule of Seven** - limit any view to no more than seven steps for a good fit.
- **Activity granularity** - activities must be similar in scope at each level. Avoid the String of Pearls pattern, that is, series of activities in the same lane.
- **Activity description** - use [action verb] + [business object] and avoid vague verbs like ‘process’ and ‘perform’.

Chapter 5. Working with self-service

IBM Cloud Orchestrator provides an intuitive Self-service user interface, where users can use the Self-Service Catalog to request resources. For example, users can deploy virtual machines, or add volumes, or manage key pairs. From this interface, users can monitor their requests and manage their resources.

About this task

IBM Cloud Orchestrator provides a rich set of predefined offerings in the Self-Service Catalog. In addition, users with the catalog editor role can create offerings with IBM Process Designer, and populate the Self-Service Catalog with additional offerings.

An offering is a Business Process Manager process in the Self-Service Catalog. For information about toolkits that you can use to build offerings, see [IBM Cloud Orchestrator content development](#).

Using self-service

Use the IBM Cloud Orchestrator Self-service user interface to request resources, monitor the status of your requests, and do additional tasks related to a resources.

Viewing the dashboard

Use the DASHBOARD tab to monitor tasks, requests, quota usage, and virtual machine status for the current project.

Inbox

The Inbox area provides an overview of the inbox statistics.

The section header displays the number of each of the following types of tasks:

- New today
- To-do (tasks that have not yet been claimed)
- Overdue

The table displays the following information about the most recent tasks:

- Latest Items
- Requested by
- Priority
- If a task is overdue, the overdue icon is displayed.

Click a task type in the section header, or click an item in the table, to open the INBOX tab.

Request History

The Request History area provides an overview of requests statistics.

The section header displays the number of each of the following types of requests:

- New today
In progress
Failed

The table displays the following information about the most recent requests:
- Latest Requests
- Submitted On
- Status

Click a request type in the section header, or click an item in the table, to open the REQUEST HISTORY tab. If you click a request type, only requests of that type are displayed.

**Quota Usage of Current Project**

The Quota Usage of Current Project area provides the current aggregate of the following items for the current project:
- vCPU Usage
- RAM (MB) Usage
- Volume (GB) Usage

Aggregates are based on all virtual machines in your project across all regions.

Usage is displayed as a percentage. The dial displays 50%, 75%, and 100% threshold indicators, which are color-coded as follows:

<table>
<thead>
<tr>
<th>Color</th>
<th>% Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>0-50</td>
</tr>
<tr>
<td>Yellow</td>
<td>51-75</td>
</tr>
<tr>
<td>Red</td>
<td>76-100</td>
</tr>
</tbody>
</table>

**VM Status**

The VM Status area provides information about the deployed virtual machines for the current project. The total number of deployed virtual machines in your project across all regions is displayed, with a breakdown based on status. The virtual machine status is color-coded as follows:

<table>
<thead>
<tr>
<th>Color</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Active</td>
</tr>
<tr>
<td>Blue</td>
<td>Paused</td>
</tr>
<tr>
<td>Red</td>
<td>Error</td>
</tr>
<tr>
<td>Yellow</td>
<td>Shutoff</td>
</tr>
</tbody>
</table>

Click a status type to open the ASSIGNED RESOURCES tab. The tab contents are filtered to display only virtual machines with the selected status.
Submitting a self-service request

Use the SELF-SERVICE CATALOG tab to view the list of offerings and submit a self-service request.

Procedure
1. Log on to the IBM Cloud Orchestrator Self-service user interface and click the SELF-SERVICE CATALOG tab.
2. Open the category of your choice to view the offerings. You can also use the Search field to look up a specific offering by name.
3. Select an offering from the list. A window with request details opens.
4. Specify any required parameters for the request.
5. Click OK to submit the request.

Results
The request is submitted. A message is displayed at the top of the page, reporting the result. You can also check the request status in the REQUEST HISTORY tab.

Viewing the status of your requests

Use the REQUEST HISTORY tab to review the status of your requests.

About this task
Users can view the progress of all actions that they submitted from the Self-Service Catalog. Administrators can view all the requests that they submitted, as well as the requests from the users that they administer.

Note: The REQUEST HISTORY tab shows only those requests that were submitted in the SELF-SERVICE CATALOG tab. Completed requests are automatically removed from the view after two weeks.

Procedure
1. Click the REQUEST HISTORY tab. All requests that you have access to are displayed on the left side of the page.
2. Click any request to view its details. The following details are displayed for a request:
   • Request ID: Unique identifier of the request.
   • Submitted on: The date when the request was submitted.
   • Submitted by: ID of the user who submitted the request.
   • Detailed information: This is a description of the request.
   • Status: The processing status of the request. The request can have one of the following statuses: New, Pending, Queued, Suspended, Running, Completing, Completed, Failing, Failed, Canceling, Canceled.
   • Status message: A message completing the status to provide more information to the user.

Note: In this view, you can only:
• Use the search box to search for items that were already loaded into the view. Any items that are not in the request list when the search is launched are not found.
Use the sort options, **Sort by Name**, **Sort by Date**, and **Sort by Status**, to sort the items that were already loaded into the view. Any items that are not in the request list when the sort is launched are not sorted.

To retrieve more list items, use the scroll.

**Managing resources**

Use the **ASSIGNED RESOURCES** tab to manage your assigned resources.

The columns for each instance type table can vary. From the table view you can launch actions on a single instance or on multiple instances. To view detailed information about an instance click anywhere on the instance row. The details screen contains actions that pertain only to the selected instances.

**Resource types**

IBM Cloud Orchestrator supports several types of resources, including domains, virtual machines, and volumes. Resources types are also known as instance types (for example, in the Core Services REST API).

IBM Cloud Orchestrator provides the following resource types:

**Action**
An action is an instance that can be applied to other instances. An action always includes a Business Process Manager process that can be run on the associated instance.

**Category**
A category is a container for self-service offerings that are displayed in the SELF-SERVICE CATALOG. You can organize your offerings inside the catalog.

**Domain**
A domain is the highest entity in the identity model. It is a container and namespace for projects and users of a customer.

**Heat**
The heat instance type represents a combined set of virtual machines created via a heat template in OpenStack.

**Offering**
An offering is an IBM Cloud Orchestrator process made available in the SELF-SERVICE CATALOG. IBM Cloud Orchestrator provides a set of offerings out of the box. However, you can create your own offerings with IBM Process Designer.

**Openstackvms**
An instance of type openstackvms represents a single OpenStack virtual server.

**Project**
A project is a container that owns resources such as virtual machines, stacks, and images.

**User**
A user represents the account of a person. You can log in to IBM Cloud Orchestrator with a user account. A user must always be member of at least one project.

**Volumes**
The volumes instance type is the disk space resource that can be attached to a virtual server.
Working with resources
You can search, sort, and apply actions to your assigned resources.

Searching in Assigned Resources:

Carry out a search for an instance based on the instance name, description, or status.

About this task
To search for a particular instance carry out the following steps:

Procedure
1. Log in to IBM Cloud Orchestrator and click the Assigned Resources tab.
2. Select an instance type from the list of entries that is displayed below the navigation menu.
3. Enter a name, description, or status of the instance you want to filter for.
4. Click enter on the search input field or click the search icon.

Results
A list of matching instances that has a name, description or status that equals the value you entered is displayed.

Sorting in Assigned Resources:

You can sort any instance type columns that have an arrow next to their name.

About this task
To sort the instance table carry out the following steps:

Procedure
1. Log onto IBM Cloud Orchestrator and click the Assigned Resources tab.
2. Select an instance type.
3. Click the column header that you want to sort.

Results
The instance type will sort itself according to the column header you choose.

Applying an action to a resource:

Use the ASSIGNED RESOURCES tab to select an action for a particular instance, for example, to start or stop a virtual machine.

Procedure
1. Log on to IBM Cloud Orchestrator and click the ASSIGNED RESOURCES tab.
2. Select an instance by selecting the single row check box. Select multiple instances by selecting the check box in the header row.
3. Select the action that you want to execute from the Actions box on the left. The following options are displayed based on the action type selected:
**Human service**

If the action you select requires a human service, a window with request details opens. Specify any required parameters for the request. Click **OK** to execute the action or **Cancel** to return to the previous view.

**No human service**

If the action you select does not require a human service, a confirmation dialog box is displayed. Select **Continue** to execute the action or **Cancel** to close the dialog and return to the main view.

**Note:** When an action is executed, a message is displayed at the top of the page, reporting the result. You can also check the request status in the REQUEST HISTORY tab.

**Related concepts:**

“Managing actions” on page 322

A Service Designer can create, edit, or delete actions in the Actions Registry, and manage the access control list for actions.

**Managing Virtual Machines**

The IBM Cloud Orchestrator Self-Service Catalog provides three Self Service Offerings to deploy virtual machines using the OpenStackNova component. It can also register and unregister a public and private key pair that can be used to access the virtual machine.

**Deploying a virtual machine:**

The IBM Cloud Orchestrator Self-Service Catalog provides a default offering that you can use to deploy a Single Virtual Server using OpenStack Nova.

**Before you begin**

- Resources such as images, flavors, keys, and networks, must be defined in the OpenStack environment. Otherwise, the virtual machine cannot be deployed.
- Images must be stored within OpenStack Glance.
- Flavors and networks must be defined in OpenStack.
- Keys must be registered with a project.
- For tasks such as processing user data or metadata, the image must be prepared to include the cloud-init package.
- Assign the region and availability zone to the project.

**Procedure**

1. Log in to the Self-service user interface as an End User.
2. In the menu bar, click SELF-SERVICE CATALOG.
3. Click **Deploy cloud services > Deploy a single virtual server**. The Deploy a single virtual server page opens.
4. Select the region where the virtual server should be deployed to.
5. Click **Next**.
6. a. Specify a **Server Name**.
   b. Select an **image**, **availability zone** and **flavor** from the drop down menus. Select the **network** by selecting the check box beside the network name.

   **Note:** There are three other options on this page that you should be aware of:
• **Use Key to Access Virtual Machine:** If you select this check box, a drop down menu appears with a list of keys. Select one of the keys to access a virtual machine.

• **Set UserId and Password:** If the checkbox is selected, three additional fields are displayed:
  – One to enter the **UserId** name.
  – One to enter the password to be set.
  – One to reenter the password to be set for validation.

Enter a **UserId** and specify whether a password is to be set on the virtual machine. To set the password on the virtual machine, the cloud-init package must be installed on the image used. Another checkbox is displayed to specify whether the password set has to be changed on first login or not.

• **Attach Volume:** If you click **Attach Volume**, a new window appears where you can select volumes from a list of volumes to be attached to the new virtual machine.

**Note:** Click the following links for more on availability zones:

• [Assigning a zone to a domain](#)
• [Modify the availability zones of a project](#)

**Note:** The default user to be used can be configured within the cloud-init config file. Each Linux distribution has its own default user. See [http://cloudinit.readthedocs.org/en/latest/topics/examples.html](http://cloudinit.readthedocs.org/en/latest/topics/examples.html) for more information.

**What to do next**

Proceed to [Managing virtual machine instances](#)

**Managing virtual machine instances:**

Virtual machine instances represent the servers (virtual machines) that are running in OpenStack backend of IBM Cloud Orchestrator.

IBM Cloud Orchestrator provides a built-in instance type called OpenStack virtual machines providing the functionality to manage deployed virtual machines. To work with virtual machines click the **ASSIGNED RESOURCES** tab in the navigation menu of IBM Cloud Orchestrator user interface and select the Virtual Machines link. The page displays a table with the following attributes for each virtual machine:

**Name**  Defines the name of the virtual machine.

**Status**  Defines the current status of the virtual machine. Possible values are **ACTIVE** for running servers or **SHUTOFF** for stopped services.

**Created with**
  The OpenStack Region the virtual machine belongs to.

**Last update**
  The time stamp of the latest change on the virtual machine.

**IP Addresses**
  The IP addresses of the virtual machines.

From this view, you can start, stop and delete virtual machines.
• **Starting one or many Virtual Machines:**
  - Select one or many virtual machines in the instances table with the status **SHUTOFF**.
  - In the **Actions** menu to the left of the table click **Start**.

  **Note:** This action is available only if all the selected virtual machines have the status **SHUTOFF**.

• **Stopping one or more Virtual Machines:**
  - Select one or more virtual machines in the instances table that are in **ACTIVE** status.
  - In the **Actions** menu to the left of the table click **Stop**.

  **Note:** This action is available only if all the selected virtual machines have the status **ACTIVE**.

• **Deleting one or more Virtual Machines:**
  - Select one or more virtual machines in the instances table that are in **ACTIVE** status.
  - In the **Actions** menu to the left of the table click **Delete**.

  **Note:** This action is available only if all the selected virtual machines have the status **ACTIVE**.

• **Executing a script**
  - Select one or more virtual machines in the instances table that are in **ACTIVE** status and also have a key pair defined.
  - In the **Actions** menu to the left of the table click **Execute Script**.
  - The first panel of **Execute Script** shows a selection box to select the network interface to be used for the script execution. Click **OK**. The second panel where you can define the script parameters appears.
  - Fill in the following script parameters:
    - **Script Repository Path**.
    - The **Script Repository Sub Folder** specifies a subfolder under the Script Repository where the script is located. If blank, the Script Repository Path is used.
    - **Script Name**.
    - **Secure Shell** user.
    - **Destination Directory**.
    - **Working Directory**.
    - **Command Line**.

  **Note:** The Execute Script action is implemented using the SCOrchestrator_Scripting_Utilities Toolkits.

  **Note:** This action is only available if the **ssh-key** is used to access the virtual machine which is specified during deployment.

  - Click **OK**.

**Related tasks:**

“Applying an action to a resource” on page 307

Use the **ASSIGNED RESOURCES** tab to select an action for a particular instance, for example, to start or stop a virtual machine.

**Related reference:**
Managing Virtual Machines

The IBM Cloud Orchestrator Self-Service Catalog provides three Self Service Offerings to deploy virtual machines using the OpenStackNova component. It can also register and unregister a public and private key pair that can be used to access the virtual machine.

Managing Heat stacks

You can deploy an existing Heat template, and then manage the Heat stack instance.

Deploying a Heat stack:

The IBM Cloud Orchestrator Self-Service Catalog provides a default offering that you can use to deploy an OpenStack Heat stack.

Before you begin

The Heat template to be deployed must be a valid Heat Orchestration Template (HOT), as defined in the OpenStack HOT specification.

All resources that are referenced in the Heat template, such as images, flavors, keys, and networks, must be defined in the OpenStack environment. Otherwise, the stack cannot be deployed.

- Images must be stored within OpenStack Glance.
- Flavors and networks must be defined in OpenStack.
- Keys must be registered with a project.

For certain tasks, such as processing user data or metadata, the image must be prepared to include the cloud-init package and the heat-cfntools package. For more information about building images for use with Heat, see “Building JEOS images for use with Heat” in the OpenStack documentation.

Procedure

1. Log in to the Self-service user interface as an End User.
2. Click SELF-SERVICE CATALOG.
3. Click Deploy cloud services > Deploy a cloud service using stack. The Deploy a Cloud Service Using Stack page opens.
4. For the template source, specify Direct Input.
5. In the Enter Heat Template field, paste the template text.

   Important: OpenStack HOT templates are sensitive to formatting issues. To avoid template validation errors, use the correct indentation.

7. In the Stack Name field, specify the name of the Heat stack instance to be deployed.
8. Specify the timeout value for the deployment. If the stack is not deployed within the specified time, an error message is displayed.
9. Optional: If you want to roll back the Heat instance if the stack fails to deploy, select the Rollback on failure check box.
10. If the template contains parameter definitions, each parameter name, description, and value is listed in the Parameters table. For each parameter, specify the parameter value:
• Default parameter values might be provided in the parameter definition in the template.

• If a parameter description is prefixed with the name of a supported lookup annotation, you can select the parameter value from a list in the Select Value column.

• Otherwise, you must type the parameter value in a field in the Enter Value column.

Tip: The template developer can use a lookup annotation to generate a list of possible values for a parameter, which helps the user to select a valid parameter value. The following lookup annotations are supported in IBM Cloud Orchestrator V2.4:

SCOIMAGE
Lookup of images from the image repository for the region.

SCOFLAVOR
Lookup of flavor size from the selection available in the region.

SCONETWORK
Lookup of available networks in the region.

SCOKEY
Lookup of the registered keys for the project.

11. Optional: To modify the Heat stack resources, click Stack Details:
   a. Select the resource that you want to modify.
   b. To view details of the volumes that are attached to the selected resource, click Volumes. To attach a volume to the selected resource, click Add Volume, specify the volume size and mount point, and click OK.
   c. To view details of the networks that are attached to the selected resource, click Network Interfaces. To attach a network to the selected resource, click Add Network Interface, specify the network name and fixed IP address, and click OK.
   d. To return to the Launch Heat Template page, click OK.

12. Click OK. A REST call is posted to the OpenStack Heat engine, and the Heat template is deployed.

13. Monitor the status of your deployment request, as described in “Viewing the status of your requests” on page 305.

Tip: If a problem occurs while you are deploying a Heat template, check the following log files for detailed information:

• The following file is on the Business Process Manager server:
  /opt/ibm/BPM/v8.5/profiles/Node1Profile/logs/SingleClusterMember1/SystemOut.log

• The following files are on the virtual machine where Heat is installed:
  /var/log/heat/api.log
  /var/log/heat/engine.log

Example

The following example is a simple Heat template to deploy a single virtual system that is based on the cirros-0.3.1-x86_64 image:

heat_template_version: 2013-05-23

description: Simple template to deploy a single compute instance

resources:
my_instance:
  type: OS::Nova::Server
  properties:
    image: cirros-0.3.1-x86_64
    flavor: m1.tiny

The following example is a simple Heat template to deploy a stack with two virtual machine instances, by using lookup annotations for parameters:

heat_template_version: 2013-05-23

description: Simple template to deploy a stack with two virtual machine instances

parameters:
  image_name_1:
    type: string
    label: Image Name
    description: SCOIMAGE Please specify image name for instance1
    default: cirros-0.3.1-x86_64
  image_name_2:
    type: string
    label: Image Name
    description: SCOIMAGE Please specify image name for instance2
    default: cirros-0.3.1-x86_64
  network_id:
    type: string
    label: Network ID
    description: SCONETWORK network to be used for compute instance

resources:
  my_instance1:
    Type: OS::Nova::Server
    properties:
      image: { get_param: image_name_1 }
      flavor: m1.small
      networks:
        - network: { get_param: network_id }
  my_instance2:
    Type: OS::Nova::Server
    properties:
      image: { get_param: image_name_2 }
      flavor: m1.tiny
      networks:
        - network: { get_param: network_id }

The following example is a simple Heat template to set the admin password for a virtual machine by using the user_data section:

heat_template_version: 2013-05-23

description: Simple template to set the admin password for a virtual machine

parameters:
  key_name:
    type: string
    label: Key Name
    description: SCOKEY Name of key-pair to be used for compute instance
  image_name:
    type: string
    label: Image Name
    description: SCOIMAGE Name of image to be used for compute instance
  password:
    type: string
    label: password
    description: admin password
    hidden: true
resources:
my_instance:
  type: OS::Nova::Server
  properties:
    key_name: { get_param: key_name }
    admin_user: sampleuser
    image: { get_param: image_name }
    flavor: m1.small
  user_data:
    str_replace:
      template:
        #!/bin/bash
        echo "Setting password to " $password
        echo $password | passwd --stdin sampleuser

    params:
      $password: { get_param: password }

What to do next

Proceed to "Managing Heat stack instances."

Related information:

- OpenStack Heat Orchestration Template (HOT) Specification
- OpenStack Heat Orchestration Template (HOT) Guide
- OpenStack Building JEOS images for use with Heat

Managing Heat stack instances:

You can use the Self-service user interface to manage deployed Heat stack instances.

Procedure

1. Log in to the Self-service user interface as an End User.
2. Click ASSIGNED RESOURCES > Stacks.
   The page shows a list of deployed Heat stack instances, with an Actions menu to the left of the list.
   The following information is displayed for each stack instance: the name, status, and description of the stack instance; the region where the stack instance is deployed; and the date when the stack instance was last updated.
   If you select one or more stack instances in the instance list, the Actions menu is updated to show only the actions that you can apply to the selected stack instances.
3. To delete an instance:
   a. Select the instance in the instance list.
   b. In the Actions menu, click Delete Stack.
4. To show more details about a stack instance, click the stack instance name in the instance list.
   The Heat Stack Details page shows the stack id, name and description; the stack status and status description; and the date that the stack instance was created. The details page also displays a list of the virtual machine instances that are associated with the stack instance, including the name, id, and status of each virtual machine instance. The Actions menu is updated to show only the actions that you can apply to the selected stack instance.
Managing key pairs
You can use offerings in the SELF-SERVICE CATALOG to manage key pairs.

Registering a key pair:
The SELF-SERVICE CATALOG provides a self-service offering to register a public or private key pair in the context of a project. The key can be used to access virtual machines securely without using userid and password. All users of the project can see and use this key pair.

Procedure
1. Log in to the Self-service user interface as a Cloud Administrator.
2. In the menu bar, click Self-Service Catalog. The self-service menu appears.
3. Click Infrastructure management > Register a key to enable access to virtual servers.
4. Enter a Project name, Key Name, Public Key and Private key in the fields provided.
   
   Note: If you click Generate Public And Private Key, a public or private key pair is generated and displayed in the corresponding fields of the panel.
5. Click OK.

Results
A message appears indicating that a key was registered successfully.

Unregistering a key pair:
The SELF-SERVICE CATALOG provides a self-service offering to unregister a public or private key pair defined in the context of a project.

About this task
You cannot unregister a key pair if a virtual machine exists that has the key pair defined.

Procedure
1. Log in to the Self-service user interface as a Cloud Administrator.
2. In the menu bar, click Self-Service Catalog. The self-service menu appears.
3. Click Infrastructure management > Unregister Key.
4. Enter a name for the project.
5. A table shows all the key pairs defined in the context of the project. Select one or multiple key pairs to be unregistered by selecting the check box beside the key pair.
6. Click OK.
Results

A message appears indicating that the key pairs you selected are now unregistered and no longer available to be selected during provisioning of a virtual machine.

Viewing the status of actions
Use the ACTION LOG tab to review the status of your actions.

About this task

You can view the progress of all actions that you have submitted from the resources and administration actions. Administrators can view all the requests that they submitted, as well as the requests from the users that they administer.

Note: The ACTION LOG tab shows requests that were submitted in the ADMINISTRATION tab and the ASSIGNED RESOURCES tab.

Procedure

1. Click the ACTION LOG tab. All requests that you have access to are displayed on the left side of the page.
2. Click any request to view its details. The following details are displayed for an action:
   - Name: Name of the action
   - Action ID: Unique identifier of the action.
   - Submitted on: The date when the action was submitted.
   - Submitted by: ID of the user who submitted the action.
   - Detailed information: This is a description of the action.
   - Status: The processing status of the action. The action can have one of the following statuses:
     - New
     - Pending
     - Queued
     - Suspended
     - Running
     - Completing
     - Completed
     - Failing
     - Failed
     - Canceling
     - Canceled
   - Status message: A message indicating why the action reached the status it is in.

Note: In this view, you can only:
- Use the search box to search for items that were already loaded into the view. Any items that are not in the request list when the search is launched are not found.
Use the sort options, **Sort by Name**, **Sort by Date**, and **Sort by Status**, to sort the items that were already loaded into the view. Any items that are not in the request list when the sort is launched are not sorted.

To retrieve more list items, use the scroll.

**Working with patterns**

Use the **PATTERNS** tab to manage virtual applications, virtual systems, and shared services.

For more information about working with patterns, see [Chapter 7, “Managing and deploying virtual patterns,” on page 343](#).

**Managing the Inbox**

Use the **INBOX** tab to view and process pending tasks and approvals.

**Viewing the Inbox**

The Inbox lists the tasks and approvals that are currently waiting to be claimed.

To view the Inbox, complete the following steps:
1. Log in to the Self-service user interface as an End User.
2. Click the **INBOX** tab.

If new tasks or approvals are created, the notification number increases and is displayed in the **INBOX** tab. The notifications are updated every minute. The **INBOX** notifications have three common states:

- The notifications number increases when:
  - A user clicks **Reassign Back** on one or many of the tasks or approvals.
  - New tasks or approvals are generated. After approximately 1 minute, the system sees these unassigned tasks and approvals and increases the notifications number.

*Note:* If there are more than 100 approval requests waiting to be claimed, the notifications number is displayed as **100+**.

- The notifications number decreases when:
  - A user claims one or many of the tasks or approvals. After approximately 1 minute, the system sees that the tasks or approvals have been claimed and decreases the notifications number.

- No notifications number is displayed when:
  - There are no tasks waiting to be claimed by the user.

**Processing an Inbox assignment**

When a submitted request requires any user interaction, such as an approval, or providing additional parameters, the responsible users get an assignment in their Inbox.

**Procedure**

1. Log on to IBM Cloud Orchestrator Self-service user interface and click the **INBOX** tab. A list of assignments that require user interaction is displayed.
   There are the following types of assignments:
Approval request

General task
You can click on the assignment icon to see details about it.

2. The assignment can have one of the following statuses:
   - If the assignment was still not claimed by any user, the Claim button is displayed. Click Claim to take the ownership of the assignment.
   - If you already claimed the assignment, the Reassign Back button is displayed. Click Reassign Back to release the assignment and allow another user to claim it.

3. To complete an assignment that you claimed, perform the following steps:
   a. Click on the assignment icon to view the assignment details.
   b. If you want to complete a general task, enter any information required and click Submit.
   c. If you want to complete an approval request, click Accept or Reject. You can optionally enter a reason.

   A completion message is displayed and the assignment is deleted from the INBOX tab.

Managing self-service offerings

A Service Designer can create self-service offerings and use them to customize the IBM Cloud Orchestrator environment. A Service Designer is a user with the catalogeditor role.

Related concepts:
IBM Cloud Orchestrator content development
IBM Cloud Orchestrator content is a set of automation packages to enable IBM Cloud Orchestrator to use the features that are delivered by external software and infrastructure devices.

Self-Service Catalog default contents
IBM Cloud Orchestrator provides a set of default offerings and categories in the Self-Service Catalog to help you to manage instances. You can modify the catalog to add, modify, and remove offerings according to your needs.

IBM Cloud Orchestrator provides offerings in the following categories in the Self-Service Catalog:

Design and create cloud service offerings
   Help to create and publish managed cloud services.

Deploy Cloud Services
   Deploys single virtual servers and cloud services relying on Workload Deployer based patterns or heat templates, including the capability to add software at deployment time.

Virtual System Operations (Single Instances)
   Manages single instances that are deployed using Workload Deployer component. This includes activities like start, restart, stop, change the
expiration date of an instance, deploy and delete instances as well as change the limit of environment profiles.

**Infrastructure management**
A set of offerings that allows you to manage secure shell keys and integrate your solution with IBM Endpoint Manager.

**Create resources for cloud services**
Allows you to manage additional volumes.

**Virtual System Operations (Multiple Instances)**
Manages multiple instances in single offerings that is start, stop, restart and delete. Like Virtual System Operations (Single Instances), this is done using Workload Deployer component.

**Managing offerings**
A Service Designer can add, edit, or delete offerings in the Self-Service Catalog, and manage the access control list for offerings.

**Creating an offering**
You can create a new offering in a domain.

**Procedure**
1. Log in to the Self-service user interface as a Service Designer.  
2. In the navigation menu, click CONFIGURATION > Self-Service Catalog.  
3. Click Offerings in the menu below the navigation menu.  
4. Click Create Offering in the Actions menu. The Create Offering window is displayed.  
5. Enter a name for the offering.  
6. Select an icon and a category for the offering.  
7. Optional: Enter a description for the offering.  
8. Select a process, application and human service for the offering.

**Note:** To find the process, select the application to filter the processes by that application. Once the process is found, select the user interface from the list of available human services for the selected process. Configure the access control. By default any user in the same domain can use the offering. The Domain Administrator and catalog editor of the domain are allowed to modify the offering.  
9. Click Create.

**Results**
A message appears indicating that the offering is created successfully.

**Editing an offering**
You can modify one or multiple offerings in a domain.

**Procedure**
1. Log in to the Self-service user interface as a Service Designer.  
2. In the navigation menu, click CONFIGURATION > Self-service catalog.  
3. Click Offerings from the list of entries that appears below the navigation menu.  
4. Click on the offering that you want to edit.
5. Click **Edit Offering** in the **Actions** menu.
6. Specify an **Icon**, **Category** and **Process** for the offering from each of the drop down menus.
7. Optional: Enter a name and description for the offering and choose application and human service for the offering from the drop down menus.
8. Click **OK**.

**Results**

A window appears in the top right of the screen confirming that the offering has been modified.

**Deleting an offering**

You can delete one or multiple offerings in a domain.

**Procedure**

1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click **CONFIGURATION** > **Self-service catalog**.
3. Click **Offerings** from the list of entries that appears below the navigation menu.
4. Click on the offering that you want to delete.
5. Click **Delete Offering** in the Actions menu.
6. Click **Confirm** from the window that opens.

**Results**

A window appears in the top right of the screen confirming that the offering has been deleted.

**Modifying access control of an offering**

You can modify the access control list of an offering by adding or removing access.

**Procedure**

1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click **CONFIGURATION** > **Self-Service Catalog**.
3. Select an offering and click **Modify Access Control List** in the **Actions** menu.
   The **Modify Access Control List** window appears displaying a table describing which domain, project and role has access rights to the action.
4. Complete one of the following options in the table:
   - To create a new entry on the table, specify the domain, project and role for a new access control entry and click **Add to Access Control List**.
   - Click on the red **X** in the table to remove an access control entry from the list.
5. Click **Save**.
Managing categories

A Service Designer can manage add, edit, or delete categories in the Self-Service Catalog.

Creating a category
You can create a new category in a domain.

Procedure
1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click CONFIGURATION > Self-service catalog.
3. Click Categories in the menu below the navigation menu.
4. Click Create Category in the Actions menu. The Create Category window is displayed.
5. Enter a name for the category.
6. Select an icon for the category.
7. Enter a description for the category.
8. Click Create.

Results
A message appears indicating that the category is created successfully.

Editing a category
You can modify one or multiple categories in a domain.

Procedure
1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click CONFIGURATION > Self-service catalog.
3. Click Categories from the list of entries that appears below the navigation menu.
4. Click on the category that you want to edit.
5. Click Edit Category in the Actions menu.
6. Specify an Icon for the category from the drop down menu
7. Optional: Enter a name and description for the category
8. Click OK.

Results
A window appears in the top right of the screen confirming that the category has been modified.

Deleting a category
You can delete one or multiple categories in a domain.

Procedure
1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click CONFIGURATION > Self-service catalog.
3. Click Categories in the menu below the navigation menu.
4. Click on the category that you want to delete.
5. Click Delete Category in the Actions menu.
6. Click **Confirm** from the window that opens

**Results**

A window appears in the top right of the screen confirming that the category has been deleted.

**Managing actions**

A Service Designer can create, edit, or delete actions in the Actions Registry, and manage the access control list for actions.

Click **CONFIGURATION > Actions Registry** in the navigation menu to manage actions. You can search for a particular instance by specifying the instance name or description in the search field. The instance table can be sorted using any column that has the sort icon.

To manage actions on Virtual System (Classic) Pattern instances, see User actions.

**Creating an action**

You can create a new action in a domain.

**Procedure**

1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click **CONFIGURATION > Actions Registry**.
3. Click **Create Action** in the **Actions** menu. The **Create Action** window is displayed.
4. Enter a name for the action.
5. Select an icon and process for the action.
6. Optional: Enter a description for the action. Select the type of instance the action applies to including the tags you want the action to apply to. Select an application and human service for the action.

   **Note:** You must specify which instance the action applies to. Based on the selection of the type, choose from a list of tags that the instance could have. The action only appears on instances having the type and tag. Specify whether the action is able to:
   - create an instance, `createInstanceAction`
   - modify only a single instance, `singleInstanceAction` or
   - modify multiple instances, `multiInstanceAction`

   Select the application to filter the processes by that application. Once the process has been found, select the user interface from the list of available human services for the selected process. Then, configure the access control. The Domain Administrator and catalog editor of the domain are allowed to modify the offering.
7. Click **Create**.

**Results**

A message appears indicating that the action is created successfully.
Editing an action
You can edit one or multiple actions in a domain.

Procedure
1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click CONFIGURATION > Actions Registry.
3. Select the check box next to the action you want displayed in the list.
4. Click Edit Action in the Actions menu.
5. Specify an Icon and a Process for the action from each of the drop down menus.
6. Optional: Enter a name and description for the action. Select the type of instance and the tags the action applies to using the drop down menus and check boxes respectively. Select Application and Human Service from the drop down menus.
7. Click OK.

Results
A window appears in the top right of the screen confirming that the action has been modified.

Deleting an action
You can delete one or multiple actions in a domain.

Procedure
1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click CONFIGURATION > Actions Registry.
3. Select the check box next to the action you want displayed in the list.
4. Click Delete Action in the Actions menu.
5. Click Confirm in the window that opens.

Results
A window appears in the top right of the screen confirming that the action has been deleted.

Modifying access control list of an action
You can modify the access control list of an action by adding or removing access.

Procedure
1. Log in to the Self-service user interface as a Service Designer.
2. In the navigation menu, click CONFIGURATION > Actions Registry.
3. Click Modify Access Control List in the Actions menu. The Modify Access Control List window appears displaying a table describing which domain, project and role has access rights to the action.
4. Complete one of the following options in the table:
   • To create a new entry on the table, specify the domain, project and role for a new access control entry and click Add to Access Control List.
   • Click on the red X in the table to remove an access control entry from the list.
5. Click **Save**.
Chapter 6. Managing virtual images

Virtual images provide the operating system and product binary files that are required to create a virtual system instance.

IBM Cloud Orchestrator adopts a modular image creation model to support different deployment scenarios. A base image that can be deployed via IBM Cloud Orchestrator is an image that can be deployed via OpenStack. For this reason, such images are also described as OpenStack-ready images. For more information about base images, see "Creating base images."

This kind of image is suitable for single instance deployments and deployment of OpenStack Heat stacks.

To run more complex scenarios in which you might add software to the instance at deployment time, or embed scalability policies, the base image must be enriched with additional software, and must adhere to some specific prerequisites. For additional information, see "Creating images to deploy via virtual system patterns or virtual application patterns" on page 330.

Images used in SmartCloud Orchestrator V2.3 can be used in IBM Cloud Orchestrator for deployment with Virtual System Pattern (Classic).

Note:
- CentOS is supported only for Virtual System Pattern (Classic) when the images are prepared using the Image Construction and Composition Tool tool that was provided with SmartCloud Orchestrator V2.3.
- Images for Linux on System z cannot be used for Virtual System Pattern (Classic). To deploy a z/VM pattern, you must use the generic OpenStack Image - Linux [s390x] image to create a Virtual System Pattern.
- Images for Linux on System z that are purchased from IBM Cloud Orchestrator Catalog are not supported in IBM Cloud Orchestrator V2.4.

Virtual System Pattern (Classic), Virtual System Pattern, and Virtual System Application can also be used with images that are purchased from IBM Cloud Orchestrator Catalog, with the exception of images for Linux on System z.

IBM Cloud Orchestrator also provides a procedure for advanced users to create SmartCloud Orchestrator 2.3-like images to use in Virtual System Pattern (Classic). For additional information see "Creating SmartCloud Orchestrator V2.3-like images (advanced users only)" on page 341.

Creating base images

You can create base images.

To create images suitable for single instance deployments and deployment of OpenStack Heat stacks, follow the instructions provided in Create images manually of the OpenStack Virtual Machine Image Guide. This type of images cannot be used for Virtual System Patterns, Virtual Application Patterns or Virtual System Patterns (classic).
The list of supported operating systems is dictated by OpenStack.

After the operating system is installed, you must install the cloud-init software for customizing the instances deployed from the image just created as follows:

- “Adding cloud-init to Linux images”
- “Adding cloud-init to Windows images” on page 327
- “Adding cloud-init to images for Linux on Power” on page 329
- “Adding cloud-init to images for Linux on System z” on page 329


Notes:

- When you create a Linux image, ensure you create a single ext3 or ext4 partition (not managed by LVM), otherwise you might have issues in installing cloud-init (see the steps below).
- If you use a KVM template, ensure that the image has a single disk; you can add additional disks at deployment time. For additional information, see “Add-ons in the catalog” on page 385.
- When you create a Linux image (any architecture), ensure that the configuration of the virtuser account either is not created on the image (it is created automatically) or, if it exists, the virtuser user is member of the virtuser group.

To use these images as part of Heat templates it is recommended to install the heat-cfn tools (for additional information see [https://wiki.openstack.org/wiki/Heat/ApplicationDeployment](https://wiki.openstack.org/wiki/Heat/ApplicationDeployment)).

**Note:** heat-cfn tools for Linux on Power can be downloaded from [http://dl.fedoraproject.org/pub/epel/6Server/ppc64/](http://dl.fedoraproject.org/pub/epel/6Server/ppc64/) for PowerVC images.

cloud-init is not supported on AIX. You can perform basic deployments only (such as plain operating system plus IP address assignment) unless you use Virtual System Patterns (classic), Virtual System Patterns or Virtual Application Patterns.

cfn tools are not supported on Windows, AIX and Linux on System z.

### Adding cloud-init to Linux images

You can add cloud-init to Linux operating system images.

To add cloud-init to your image set up a YUM repository that points to EPEL 6 software in [https://fedoraproject.org/wiki/EPEL](https://fedoraproject.org/wiki/EPEL) then run the following commands:

```bash
yum install cloud-init
yum install cloud-utils
yum install dracut-modules-growroot
```

For more information, see OpenStack Linux image requirements.
Adding cloud-init to Windows images

You can add cloud-init to Windows operating system images.

To add cloud-init to your image, download it from https://www.cloudbase.it/downloads/CloudbaseInitSetup_Beta.msi.

Then install cloudbase-init by following the procedure at http://www.cloudbase.it/cloud-init-for-windows-instances/.

Note:
- After the cloudbase-init installation, do not select the option to run sysprep.exe in the Finish page.
- When you create the network, set the dns-nameservers and gateway parameters.
- To speed up the IP address injection, when you create the image template, specify the metadata_services parameter in the cloudbase-init.conf file:

  metadata_services= 'cloudbaseinit.metadata.services.configdrive.ConfigDriveService,
  cloudbaseinit.metadata.services.httpservice.HttpService,
  cloudbaseinit.metadata.services.ec2service.EC2Service,
  cloudbaseinit.metadata.services.maasservice.MaaSHttpService'

- If you get the OS can not be restarted automatically message after changing the host name, use the latest cloudbase-init version.
- cloudbase-init allows to set password for a user. The user name is configured at image preparation time and cannot be modified at virtual machine creation time. You can specify a user name during cloudbase-init installation or in the cloudbase-init.conf file. If the user does not exist, a new user account is created at virtual machine initialization time. If there are multiple Windows users at image preparation time, at virtual machine initialization time password is changed only for the user specified in the cloudbase-init configuration. Other user's passwords are not changed.

After cloudbase-init is installed, complete the following procedures.

Installing virtio driver (KVM hypervisor only)

To use Windows operating system images on a KVM hypervisor, install the virtio driver into the system because OpenStack presents the disk using a VIRTIO interface while launching the instance.

You can download an virtio-win*.iso file containing the VIRTIO drivers from the following location: http://alt.fedoraproject.org/pub/alt/virtio-win/latest/images/bin/

Use virt-manager to connect virtio-win*.iso to the image and update the network adapter in the virtual machine by completing the following steps:

1. Right-click Computer > Properties > Change settings > Hardware > Device Manager.
2. Click Network adapter > Update driver software > Browse my computer for driver software.
3. Select the virtual CD/DVD drive and then select the inf file.
4. Restart the virtual machine.
Running sysprep.exe

Run sysprep.exe to remove all the unique system information, like computer name and hardware specific information, from your Windows image.

To run sysprep.exe on Windows 2008 R2, complete the following steps. Refer to the Microsoft documentation for the other Windows platforms.


2. Copy the install.wim file from the \sources directory of the Windows 2008 R2 installation DVD to the hard disk of the virtual machine.

3. Start the Windows System Image Manager.

4. In the Windows Image pane, right-click Select a Windows image or catalog file to load the install.wim file you just copied.

5. When a warning that the catalog file cannot be opened is displayed, click Yes to create a new catalog file. Remember to select the Windows 2008 R2 Edition.

6. In the Answer File pane, right-click to create a new answer file:
   Language and Country or Region:
   a. Generate the answer file from the Windows System Image Manager by expanding Components in your Windows Image pane, right-click and add the Microsoft-Windows-International-Core setting to Pass 7 oobeSystem.
   b. In your Answer File pane, configure the InputLocale, SystemLocale, UILanguage, and UserLocale with the appropriate settings for your language and country or region.

   Administrator Password:
   • In the Windows Image panel, expand the Microsoft-Windows-Shell-Setup component, and expand User Accounts, right-click on AdministratorPassword, and add the setting to the Pass 7 oobeSystem configuration pass of your answer file.
   • In the Answer File panel, specify a password next to Value.

   Note: You can read the AIK documentation and set more options depending on your deployment. The steps described here are the minimum needed for the Windows unattended setup.

   Software License Terms:
   In the Windows Image panel, expand Components and find the Microsoft-Windows-Shell-Setup component. Highlight the OOBE setting, and add the setting to the Pass 7 oobeSystem. In the Answer File panel, set HideEULAPage true in OOBE settings.

   Product Key and Computer Name:
   • In the Windows Image panel, right-click on the Microsoft-Windows-Shell-Setup component and add the settings to the Pass 4 specialize configuration pass of your answer file.
   • In the Answer File panel, enter your Product Key in the space provided next to ProductKey. Furthermore, to automate the Computer Name Selection page, specify a computer name next to ComputerName.

7. Save the answer file as unattend.xml. Ignore the warning messages that appear in the validation window.
8. Copy the unattend.xml file into the c:\windows\system32\sysprep directory of the Windows 2008 R2 Virtual Machine.

9. Clean the environment of the virtual machine.

10. Uninstall Windows AIK which might not be part of the virtual machine you create.

11. Remove the install.wim file that was copied to the virtual machine.

12. Run the sysprep tool as follows:
   
   ```
   cd c:\Windows\System32\sysprep
   sysprep.exe /oobe /generalize /shutdown
   ```

   The Windows 2008 R2 virtual machine shuts down automatically after sysprep is complete.

**Adding cloud-init to images for Linux on Power**

You can add cloud-init to images for Linux on Power.

For information about how to add cloud-init to images for Linux on Power, see

**Adding cloud-init to images for Linux on System z**

You can add cloud-init to images for Linux on System z.

Specific information on how to build z/VM images is available in chapter 6 of the Enabling z/VM for OpenStack guide at http://www.vm.ibm.com/sysman/openstk.html

**Note:** Reversal of the order of the services startup for z/VM: sshd must be run before cloud-init.

Service startup order is determined by /etc/rc.d/rc[0-6].d/ directory which contains symlinks to /etc/init.d/ files. To change startup order, in particular runlevel, you must change the names of files in the appropriate directory. Scripts are run by name order so that script with lower number behind letter S starts start earlier. In z/VM sshd needs to start before cloud-init so it must have a lower number. Changes must be done for at least the default runlevel, which in RHEL is 3.

To update the RHEL image to make sure cloud-init-local starts after sshd service is started, perform the following steps:

1. In /etc/init.d/cloud-init-local file, add sshd in the Required-Start statement:
   
   ```
   # Required-Start: $local_fs $remote_fs xcatconf4z sshd
   ```

2. From the command line, run following commands
   
   ```
   chkconfig cloud-init-local off
   chkconfig cloud-init-local on
   ```
Creating images to deploy via virtual system patterns or virtual application patterns

You can create an image that can be used for virtual systems patterns or virtual application patterns.

Choose one of the following options:
- Use a base image as described in “Creating base images” on page 325.
- Use a SmartCloud Orchestrator V2.3 image.

The list of supported platforms shrinks with respect to the plain OpenStack images. For more information, see table 2 in “Software prerequisites” on page 21.

Important: Although OpenStack supports having multiple images in Glance with the same name in the same region, do not use the same image name in different regions for images that are not logical copies. Doing so might result in the wrong image being used at deployment time.

SmartCloud Orchestrator V2.3 images can be used for deploying Virtual System (Classic) Patterns.

Whether you are using a base image or a SmartCloud Orchestrator V2.3 image, you must install additional software on the image and run some operating system configuration as explained in the following topics:

Software prerequisites for Linux images (KVM or VMware hypervisors)

Ensure that the software requirements are met before you create a Linux image.

To create the image, see “Creating base images” on page 325 or use a SmartCloud Orchestrator V2.3 image.

The following software must be installed:
- curl
- python
- sed
- dos2unix
- openssl
- ksh
- compat-libstdc++
- glibc.i686
- libgcc.i686
- compat-libstdc++-33.i686
- nss-softokn-freebl.i686

The following python modules must be installed:
- base64
- gettext
- hashlib
- pycurl
Software prerequisites for Microsoft Windows images

Ensure that the software requirements are met before you create a Microsoft Windows image.

To create the image, see “Creating base images” on page 325 or use a SmartCloud Orchestrator V2.3 image.

The following software must be installed:

- Microsoft Visual C++ 2008 Redistributable (x64)
- OpenSLL (for example, Win64OpenSSL-1_0_1g.exe)
- Curl 32bit (for example, curl-7.36.0-win32-fix1.msi)
- Python 2.X 32bit (for example, python-2.7.6.msi)
- Python module: pywin32 32bit (for example, pywin32-218.win32-py2.7.exe)
- Python module: pycurl 32bit (for example, pycurl-7.19.3.1.win32-py2.7.msi)

**Important:** You must install VMware Tools after Microsoft Visual C++ 2008 Redistributable is installed. If VMware Tools are already installed, you must uninstall and reinstall VMware Tools after Microsoft Visual C++ 2008 Redistributable is installed.

Ensure that you disable the Microsoft Windows User Account Control (UAC).

Software prerequisites for images for Linux on System z

Ensure that the software requirements are met before you create an image for Linux on System z.

To create the image, see “Creating base images” on page 325.

The following software must be installed:

- gcc
- make
- gtk2.s390
- libXtst.s390
- libstdc++.so.5
- compat-libstdc++-33.s390x
- dos2unix.s390x
- ksh.s390x
- genisoimage.s390x
- python
- curl

**Note:** For SUSE Linux Enterprise Server, use following replacements:

- For libXtst.s390, use xorg-x11-libs
- For compat-libstdc++, use libstdc++33

The following python modules must be installed:

- base64
- gettext
Software prerequisites for AIX images

Ensure that the software requirements are met before you create an AIX image.

To create the image, see “Creating base images” on page 325 or use a SmartCloud Orchestrator V2.3 image.

The following software must be installed:
- rpm.rte
- gcc-4.2.0-3.aix6.1.ppc.rpm
- activation engine

You can download the rpm.rte fileset from ftp://public.dhe.ibm.com/aix/freeSoftware/aixtoolbox/INSTALLP/ppc/rpm.rte

To install it, run the following command:
installp -d <absolute_path_of_the_package> /rpm.rte -acgXY rpm.rte


Adding images to your OpenStack environment

You can add images to your OpenStack environment to be used by IBM Cloud Orchestrator.

About this task

To use an image in IBM Cloud Orchestrator, you must add the image to your OpenStack environment.

If you are using Linux on System z, follow the instructions in Enabling z/VM for OpenStack guide at http://www.vm.ibm.com/sysman/opstk.html.

If you are using VMware, you can populate the Glance repository automatically using the discovery process (see “Configuring vmware-discovery” on page 123) or you can add images to the Glance repository manually. If you rely on VMware discovery, you can skip the remaining part of this section.

If you are using PowerVC, images are displayed in Glance automatically, without any additional action.

If you are using KVM, the only available option is the image in Glance.
To add an image to OpenStack, complete the following steps on the Region Server where OpenStack is installed:

**Procedure**

1. Set the environment by running the following command:
   ```
   source /root/keystonerc
   ```
2. Run the following command on one line:
   ```
   glance image-create
   --name image_name
   --disk-format disk_format
   --container-format container_format
   --is-public [True|False]
   < image_path
   ```

   where

   `image_name`
   
   Specifies a name for the new image that you are adding.

   `disk_format`
   
   Specifies one of the following disk formats:

   - `raw`
     An unstructured disk image format.
   - `qcow2`
     A disk format supported by the QEMU emulator that can expand dynamically and supports copy-on-write.
   - `vmdk`
     For a VMware hypervisor, another common disk format supported by many common virtual machine monitors.

   `container_format`
   
   Specifies the container format for the image. The acceptable formats are: aki, ami, ari, bare, and ovf.

   `--is-public`
   
   Specifies whether the image is accessible by other users. The value can be true or false.

   `image_path`
   
   Specifies the full path of the image to be added.

For more information about the `glance image-create` command, see the [OpenStack documentation](#)
Attention: If you are deploying on VMware, specify these additional properties:

vmware_adaptertype, vmware_ostype, and vmware_disktype.

For example:
```bash
    glance image-create
    --name my_vmware_windows_image
    --disk-format vmdk
    --container-format bare
    --is-public False
    --property vmware_disktype="preallocated"
    --property vmware_adaptertype="lsiLogic"
    --property vmware_ostype="Windows764_Guest"
    < /tmp/images_to_create
```

where vmware_disktype can be sparse|preallocated|streamOptimized, and the vmware_adaptertype can be ide|busLogic|lsiLogic. VMDK disks converted by the qemu-img utility are always monolithic sparse VMDK disks with an IDE adapter type. If the image does not come from the qemu-img utility, vmware_disktype and vmware_adaptertype might be different.

To determine the image disk type and adapter type from an image file, use the head -20 vmdk_filename command, and find the createTime and ddb.adapterType in the command output. Currently, the operating system boots VMDK disks with an IDE adapter type that cannot be attached to a virtual SCSI controller. Disks with one of the SCSI adapter types (such as busLogic or lsiLogic) cannot be attached to the IDE controller. Therefore, as the previous examples show, it is important to set the vmware_adaptertype property correctly. The default adapter type is lsiLogic, which is SCSI. You can omit the vmware_adaptertype property only if the image adapter type is lsiLogic.

When you create an image by using the glance image-create command or the Administration user interface, and the image format is not of raw type, you must specify the minimum disk space required for the image to run (that is, the current disk size). The disk size is specified in GB, and the value must be greater than or equal to the virtual size of the image. You can use the following command to find the virtual size of an image:
```bash
    # qemu-img info
```

Tip: If using the glance image-create command, specify the minimum disk size by using the --min-disk value option. If using the Administration user interface, specify the required value in the Minimum Disk (GB) field.

---

## Adding images to IBM Cloud Orchestrator

Base images can be automatically deployed via IBM Cloud Orchestrator as soon as you register them in Glance. They can be deployed as either single instances or as part of OpenStack Heat stacks.

Additional actions must taken to use images in virtual system patterns, virtual application patterns, and virtual system patterns (classic). You must use images that are created as described in "Creating images to deploy via virtual system patterns or virtual application patterns" on page 330 or OVA images that are purchased from IBM Cloud Orchestrator Catalog or images for SmartCloud Orchestrator V2.3.
Images that are created as described in “Creating images to deploy via virtual system patterns or virtual application patterns” on page 330 should not be registered in any way in the IBM Cloud Orchestrator user interface. You can select these images directly from Glance when you create the virtual system pattern.

Restriction: If such an image is registered with IBM Cloud Orchestrator, it does not deploy correctly.

To use OVA images that are purchased from IBM Cloud Orchestrator Catalog, follow the procedure “Using OVA images from IBM Cloud Orchestrator Catalog on KVM or VMware” or “Using OVA images from IBM Cloud Orchestrator Catalog on PowerVC regions” on page 338.

To use images that were created for SmartCloud Orchestrator V2.3, follow the procedure “Using images created for SmartCloud Orchestrator V2.3” on page 340.

Using OVA images from IBM Cloud Orchestrator Catalog on KVM or VMware

You can import OVA images that are purchased from IBM Cloud Orchestrator Catalog, and map the images to IBM Cloud Orchestrator.

You can purchase OVA images from IBM Cloud Orchestrator Catalog.

To import these OVA images and map them to IBM Cloud Orchestrator, complete the following steps:

1. Log in to the IBM Cloud Orchestrator UI with the admin or catalogeditor role, and from Patterns > Virtual Images and click Create New.

2. Specify the full path to the OVA file and specify the credentials if the file is on a remote system requiring authentication.

   The location can be one of the following:
   • The path to a local file Workload Deployer server
   • A remote URL on http server
   • A remote location available via SCP

3. Wait for the image import process to complete; this can take some time, depending on image size and location.

4. Accept the image license.

To add the image to OpenStack and map it to the imported virtual image:

1. Only for virtual images built on a single virtual disk, you can check out the image disk file to the OpenStack Image service (Glance). Perform the following steps:
   a. Log in to the IBM Cloud Orchestrator UI with the admin or catalogeditor role, and from Available on Locations click Manage locations.
   b. Click Create New to create a new image location and select the Perform image check out operation.
   c. Select the regions where you want to move the image and click Create. Wait for the operation to complete.

If the check out operation fails with the following error message, add the OVA image to the VMware region and map it to the Workload Deployer virtual image manually, as described in “Adding an OVA image to OpenStack” on page 336 and “Mapping the image created in IBM Cloud Orchestrator” on page 338.
CWZCA0028E: Checkout of virtual image DB2 AWSE 10.5.0.2 into region RegionKVM has failed due to the following error: This operation is not supported for templates with multiple hard disks.

2. Because Glance does not manage images with multiple virtual disks, to add this type of images to the VMware OpenStack region and then map them to IBM Cloud Orchestrator you must perform the manual procedure, as described in “Adding an OVA image to OpenStack” and “Mapping the image created in IBM Cloud Orchestrator” on page 338 respectively.

Adding an OVA image to OpenStack

Use the OVA converter tool to transform the IBM HyperVisor Edition OVA images into a format that can be used by VMware vSphere.

Before you begin

You can deploy an OVA image on VMware in two ways:

- Manually: use the VMware Open Virtualization Format Tool (the ovftool command).
- Automatically: use the OVA converter script (ibm_ova_converter.py) to invoke the ovftool command.

The ibm_ova_converter.py script is in the installation_images/utils/iwd_utils directory, where installation_images is the directory where you unpacked the IBM Cloud Orchestrator installation packages, for example, /opt/ico_install.

To run the OVA converter tool, you must have the following prerequisites installed:

- Python version 2.6 or 2.7
- Gunzip
- VMware Open Virtualization Format Tool version 3.5.2

Procedure

1. Deploy the image in either of the following ways:

   - Manually:
     a. Convert the OVA file to an OVF directory that can be used by the VMware ovftool command, by running the following command on one line:

        python ibm_ova_converter.py --file ova_path
        [--temp-dir output_dir]

        The converter script produces an OVF directory with the same name as the source OVA file, without file extension.

     b. Change directory to the OVF directory:

        cd ova_converter_output

     c. Use the VMware ovftool command to deploy the output OVF on VMware vSphere, by running the following command on one line:

        ovftool --noSSLVerify --skipManifestCheck --acceptAllEulas
        --datastore=vmware_datastore
        ova_converter_output.ovf
        "vi://vmware_user:vmware_password@vmware_host/vmware_datacenter/host/vmware_cluster/"

   - Automatically:
Convert the OVA file and deploy the image on VMware automatically, by running the following command on one line:

```bash
git lib/ova_converter.py --file ova_path
[--temp-dir output_dir]
--user vmware_user
--password vmware_password
--host vmware_host
--datacenter vmware_datacenter
--cluster vmware_cluster
--datastore vmware_datastore
[--template-name template_name]
```

where:

- `ova_path` - The full path name of the OVA file to be converted.
- `output_dir` - The target directory where the converted OVF directory is to be placed. If this optional parameter is not specified, the output is stored in the same location as the source OVA file.
- `ova_converter_output` - The OVF directory created by the OVA converter tool. The `ova_converter_output.ovf` file is located in this directory.
- `vmware_user` - The VMware user name.
- `vmware_password` - The password for the specified VMware user.
- `vmware_host` - The host name or IP address of the VMware vSphere.
- `vmware_datacenter` - The name of the target datacenter on VMware where the image is to be deployed.
- `vmware_cluster` - The name of the target cluster on VMware where the image is to be deployed.
- `vmware_datastore` - The name of the target datastore on VMware where the image is to be deployed.
- `template_name` - The template name to be used when deploying the image to VMware. If this optional parameter is not specified, the OVA file name is used.

2. After the OVF is deployed, disable vApp options in the virtual machine settings in VMware.
3. Convert the virtual machine to a template in VMware.
4. Wait for the template to be discovered in Glance.
Mapping the image created in IBM Cloud Orchestrator

You must map the image created IBM Cloud Orchestrator.

About this task

After the image is available in Glance, you must match the metadata stored in IBM Cloud Orchestrator when the image was created in the IBM Cloud Orchestrator UI, with the actual image in Glance. To do this, perform the following steps:

Procedure

1. Log in into the IBM Cloud Orchestrator UI as a user with admin or catalogeditor role.
2. Click Patterns > Virtual Images.
3. Select the image that corresponds to your .OVA file and, in Available on Locations, click Managed image locations.
4. Click Create New and select Create image mapping.
5. Select the region in the Region menu and then select the row corresponding to the image in Glance.
6. Click Create.

Results

The image is ready to be used in virtual system patterns (classic), virtual system patterns or virtual application patterns.

Using OVA images from IBM Cloud Orchestrator Catalog on PowerVC regions

You can use OVA images that are purchased from IBM Cloud Orchestrator Catalog on PowerVC regions.

Before you begin

Before you use OVA images that are purchased from IBM Cloud Orchestrator Catalog, use the tar command to uncompress the OVA files. The OVA files can include an OVF file and a mksysb image, or an OVF file and a raw disk. If the OVA file includes an OVF file and a mksysb image, you must use Network Installation Management (NIM) to restore the mksysb image to obtain a raw disk, as described in Using a mksysb image to install the base operating system on a NIM client. If the image is already in raw disk format, follow the instructions below.

Important: Some .raw files are raw images, and some .raw files are compressed files in .gz format. To identify the file type, run the file command, as shown in the following example:

```
file image1.raw
```

If the OVA file is a compressed file, you must uncompress the OVA file before the import, to make the file bootable.

Procedure

1. Copy the raw file onto a VIOS, and copy that disk image into a PowerVC volume to create and populate the disk on the SAN.
2. After the image is copied into a PowerVC volume, use the `powervc-volume-image-import` command to import it as an image:

   ```bash
   powervc-volume-image-import [-h] --name NAME --os-distro {aix, rhel, sles, ibmi} --volume VOLUME-ID [--user USER] [--activation-type {ae, cloud-init}] [--ovf OVF]
   ``

   where `--os-distro` is `aix`, `--activation-type` is `ae`, and the OVF file is the one that you extracted from the OVA file. The image is visible in Glance.


3. You can register the image, as described in “Using images created for SmartCloud Orchestrator V2.3” on page 340.

   Alternatively, you can import the OVA file into IBM Cloud Orchestrator, and then map the file to the actual disk on PowerVC, as follows:

   a. Log in to the Self-service user interface as a Service Designer.

   b. Click Patterns > Virtual Images.

   c. Click Create New.

   d. Specify the full path to the OVA file, and specify the credentials if the file is on a remote system that requires authentication. Click OK.

   e. Link the imported artifact to the disk on PowerVC, as described in “Mapping the image created in IBM Cloud Orchestrator” on page 338.

---

### Making PowerVC images compatible with Workload Deployer

Create your own images that are compatible with Workload Deployer.

To create your own image (not purchased from IBM) and make it compatible with Workload Deployer, complete the following steps:

1. Deploy an AIX image on PowerVM using your preferred method, for example, Network Installation Management (NIM).

2. Install the Workload Deployer enablement software.
   - Log in to the AIX Virtual Machine using Secure Shell and download the Workload Deployer AIX enablement package.

3. Run Visual Studio Authoring Extensions (VSAE)
   - Using the PowerVC documentation, reset VSAE and prepare the system for capture.

4. Capture the system using PowerVC:
   - From the PowerVC UI Service Panel, click manage existing and select your newly deployed Virtual Machine from the list.
   - Click capture to generate an image of the instance. The instance should already be stopped when VSAE was activated in the previous step.

5. Import the image into Workload Deployer.

6. Grab the Open Virtualization Format (OVF) from the Workload Deployer. The OVF that the Workload Deployer must apply to the image is available at the following path on the Workload Deployer Server:

   ```bash
   /drouter/ramdisk2/mnt/raid-volume/raid0/templates/rainmaker-templates/1/<image-name>.ovf
   ```

7. Transfer the OVF to the PowerVC server.

8. Edit the Glance image in PowerVC using OVF:
a. Create a PowerVC source file to allow access to keystone (/root/powervc-source) as in the following example:

```bash
export OS_USERNAME=<root user>
export OS_PASSWORD=<root password>
export OS_TENANT_NAME=ibm-default
export OS_AUTH_URL=https://<powervcserver>/powervc/openstack/identity/v2.0/
export OS_CACERT=/etc/pki/tls/certs/powervc.crt
export OS_REGION_NAME=RegionOne
```

b. Source the file:

```bash
source /root/powervc-source
```

c. Create the auth token:

```bash
keystone token-get
```

Output:

```
+-----------+----------------------------------------------------+
| Property  | Value                                              |
+-----------+----------------------------------------------------+
| expires   | 2014-08-17T12:11:47Z                               |
| id       | d7c49ff3ee37440189a47daece4ad944                   |
| tenant_id| ba1b17e3095246a80177019701e014f5                   |
| user_id  | 0                                                  |
+-----------+----------------------------------------------------+
```

Copy the id property into a file named `auth-token`

d. Get the Glance ID of the image to be used:

```bash
glance image-list
```

Use the ID, the `auth-token` file, and the `<image-name>.ovf` files to apply the OVF to the image:

```bash
python /usr/lib/python2.6/site-packages/nova/compute/ibm/configuration_strategy_ovf.py
--ovf <image-name>.ovf --auth_token auth-token --image_id <image-id> --replace
```

The image is now ready to be used by the Workload Deployer.

**Using images created for SmartCloud Orchestrator V2.3**

**About this task**

You can use images created for SmartCloud Orchestrator V2.3 only as part of virtual system patterns (classic). After they have been added to Glance, you must register them in the Self-service user interface. You can perform this action if you are in admin or catalogeditor role.

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Images**.
2. Click **Register OpenStack image** and type in the name of the image or use the lookup function to find it in Glance.
3. Select the operating system in the list and click **Register**.

**Note:** Images for Linux on System z cannot be used for virtual system patterns (classic).
Creating SmartCloud Orchestrator V2.3-like images (advanced users only)

You can create or adapt existing images to deploy as part of Virtual System Pattern (Classic). This procedure applies to Windows and Linux images for KVM or VMware hypervisors.

**Before you begin**

This is the list of the supported operating systems (see KVM and VMware rows only) [http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/c_os.html?lang=en](http://www-01.ibm.com/support/knowledgecenter/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/c_os.html?lang=en)

The image must meet the following prerequisites:

On Windows [http://www-01.ibm.com/support/knowledgecenter/api/content/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/scenarios/c_prereq_kvm_vmware_images_win.html](http://www-01.ibm.com/support/knowledgecenter/api/content/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/scenarios/c_prereq_kvm_vmware_images_win.html)

On Linux [http://www-01.ibm.com/support/knowledgecenter/api/content/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/scenarios/c_prereq_kvm_vmware_images.html](http://www-01.ibm.com/support/knowledgecenter/api/content/SS4KMC_2.3.0/com.ibm.sco.doc_2.3/scenarios/c_prereq_kvm_vmware_images.html)

**About this task**

You can add the activation engine and scp-cloud-init to the image. For details about the activation engine, see the SmartCloud Orchestrator V2.3 documentation in the IBM Knowledge Center.

You must have Administrator privileges on Windows and root privileges on Linux.

**Procedure**

2. Start an instance of the image.
3. Copy IconImageSynchronizer.zip in an instance directory and unzip it.
4. On Windows run IconImageSynchronizer.cmd and on Linux, give executable permission to IconImageSynchronizer.sh and run it.
5. Copy ovf-env.xml from the installation media, on Windows under `c:\windows\setup\ibm\AP` and on Linux under `/opt/IBM/AE/AE.AP`.
6. On Windows, run the following command:
   ```
   c:\windows\setup\ibm\AE.bat --reset -n
   ```
   On Linux, run the following command:
   ```
   /opt/IBM/AE/AE.sh --reset -n
   ```
7. Shut down the instance.
8. Convert the instance back into an image.
What to do next

To add the image to IBM Cloud Orchestrator, see "Using images created for SmartCloud Orchestrator V2.3" on page 340.
Chapter 7. Managing and deploying virtual patterns

You can perform a variety of tasks on virtual applications, virtual systems, and shared services.

Virtual pattern types

The types of virtual patterns are: virtual systems (classic), virtual systems, virtual applications, and shared services.

Virtual systems are deployed from virtual system patterns. These patterns consist of parts which are virtual machines and scripts running on those virtual machines.

Virtual system (classic)
A virtual system instance is a collection of virtual machines. Each virtual machine in a virtual system instance represents a physical node in an application server environment. For more information about virtual system patterns (classic), see “Working with virtual system patterns (classic)” on page 411.

Virtual system
A virtual system patterns (classic) with the following enhancements:

- The operating system and middleware are managed separately. You can customize an operating system image once and reuse it. Then, you can use software components to install middleware on the image.
- You can use dynamic scaling policies to define quality of service levels for software artifacts in the virtual system. Policies can be applied globally at the pattern level or specified for individual components.
- You can set versions for virtual system patterns, script packages, and add-ons to manage these assets when they change.
- Virtual system patterns are built and deployed on the same underlying architecture as virtual application patterns. You have the same control over the topology and configuration as before, but you can also build virtual system patterns and manage your deployed virtual system patterns with the tools that were previously available only for virtual application patterns, such as the Pattern Editor.

Virtual application
A virtual application is defined by a virtual application pattern. It is a complete set of platform resources that fulfill a business need, including web applications, databases, user registries, messaging services, and transaction processes. For more information about virtual application patterns, see “Working with virtual applications” on page 474.

Shared service
Shared services provide a predefined virtual application pattern that is deployed and shared by multiple application deployments in the cloud, including virtual applications, virtual systems, and virtual appliances. For more information about shared services, see “Working with shared services” on page 628.

For more information about virtual pattern types, see “Managing pattern types” on page 468.
Managing environment profiles

You can use environment profiles to control some aspects of your deployment. You can use environment profiles to group related deployment configuration options together and deploy from a single pattern.

Environment profiles overview

Environment profiles group related deployment configuration, like virtual machine names, IP address assignment, and cloud groups. Deploying patterns with environment profiles enables deployments across tiers from a single pattern.

An environment profile provides configuration that can be used when deploying a pattern. An environment can be specified with multiple clouds, and specific resources within those clouds, in IBM Cloud Orchestrator. Environment profiles provide the following function:

- Defining the operational environments, for example development, test, or quality assurance
- Defining virtual machine naming conventions within the operational environment
- Specifying whether IBM Cloud Orchestrator or the pattern deployer provides the IP address on the deployment
- Segmenting the clouds, and IP groups within the clouds, to specific environments
- Assigning aliases to the cloud resources such as clouds and IP groups
- Assigning sections within the clouds to specific users or groups

Environment profiles provide an option to deploy a pattern to a specified cloud group. You can define profile information for the cloud, IP group, and IP address at a part level in an environment profile. You can select specific IP groups for each cloud and provide aliases to the cloud and IP groups to better describe the environment at deployment time. You can use the same pattern and deploy in different environments without changing the pattern.

The virtual machine name syntax is also specific to the cloud.

Related tasks:

- **Managing environment profiles in the user interface** on page 345
  You can manage environment profiles with the IBM Cloud Orchestrator user interface from the Environment Profiles window.

Related reference:

- **Environment profiles REST API** on page 745
  You can use the representational state transfer (REST) application programming interface (API) to manage environment profiles.
Managing environment profiles in the user interface

You can manage environment profiles with the IBM Cloud Orchestrator user interface from the Environment Profiles window.

Before you begin

You must have a cloud group configured and ready, with all hypervisors configured and available to create an environment profile that is ready to be deployed.

About this task

You can use environment profiles to group related deployment configuration, like virtual machine names, IP address assignment, and cloud groups, with the IBM Cloud Orchestrator user interface.

Procedure

1. Click PATTERNS > Deployer Configuration > Environment Profiles on the menu bar to open the Environment Profiles window.
2. Work with a profile. You can perform the following tasks:
   - Create an environment profile with the information in “Creating an environment profile.”
   - Clone an existing environment profile for reuse with the information in “Cloning an environment profile” on page 349.
   - Edit an existing environment profile with the information in “Editing an environment profile” on page 350.

Creating an environment profile

You can create an environment profile with the IBM Cloud Orchestrator user interface.

Before you begin

You must have a cloud group configured and ready, with all hypervisors configured and available, to create an environment profile that is ready to be deployed.

About this task

You can create an environment profile with the steps in this task.

Procedure

1. From the upper left panel of the Environment profiles window, click New to add an environment profile.
2. Provide the following basic information about the environment profile you are creating:
   - Name Enter a unique name for the profile in the Name field. This information is required.
   - Description Optionally, enter a detailed description to identify the profile in the Description field.
Hypervisor type
Select OpenStack as the type of hypervisor in the cloud group you are using.

Environment
Select the environment in which this profile is to be created. The following options are available:
- All
- Development
- Test
- Quality Assurance
- Performance
- Research
- Production
- Pre-Production

The default value is All.

3. Click OK to create the profile. When the information is processed, you return to the Environment Profiles view and the profile you created is added to the list in the left panel. It is selected so that the information about it is shown in the right panel. For more information about the fields on this panel, see "Environment Profiles window fields" on page 351.

4. Complete the configuration. Before the environment profile is ready to use, you must provide additional configuration information in the following fields:

Virtual machine name format
It must contain one of the following variables:

$\{hostname\}$
Replaced with the host name of the virtual machine, for example: My$\{hostname\}$VM.

Note: Underscores are not valid characters in the virtual machine hostname.

$\{vs-name\}$
Replaced with the name of the virtual system instance, for example: My$\{vs-name\}$VM. This variable cannot be used alone in the Virtual machine name format field. The $\{vs-name\}$ variable must be used with one of the other formatting variables. Otherwise, if a cluster pattern is being deployed, all virtual machines would then have the same name and the deployment would fail.

$\{x-counter\}$
Replaced with a counter of $x$ digits, for example: MyVM$\{3-counter\}$. The $x$ in this example represents the number of digits for the counter. So if the value of $x$ is two, then it is represented as 02. This value could be 01, 02 or 03, for example.

IP addresses provided by
Choose whether you want the IP address for a virtual machine to be provided by IBM Cloud Orchestrator or specified when the pattern is being deployed. Use the following options:
Pattern deployer
To provide the IP address for a virtual machine at deployment, you must also specify the following information for each part:
- Cloud group
- IP group
- Host name
- IP address

Important: If you choose this option, then you cannot specify an IP address that is contained within the IP groups that are defined in IBM Cloud Orchestrator at deployment.

IP Groups
If IBM Cloud Orchestrator is to provide the IP address for a virtual machine, then you specify only the cloud group and IP group. Specify these options when you define the parts to deploy the pattern. IBM Cloud Orchestrator provides the IP address information.

Deploy to cloud groups
Click this field to select cloud groups that are configured and ready for use. Only valid cloud groups that are configured with the correct hypervisor type are available. Selecting a cloud group provides the following information for the IP groups in that cloud group:

- In use: Click this check box to use the IP group in the environment profile.
- Name: Shows the name of the IP group in the cloud you selected.
- Alias: You can specify an alias name for the IP group for use in the environment profile. The default setting is the actual name of the IP group.
- Subnet address: Shows the subnet address of the IP group.
- Gateway: Shows the gateway address of the IP group.
- Netmask: Shows the netmask address of the IP group.

Windows domain information
The Windows domain section in the environment profile is optional. If the Domain name field is empty, other fields in the section will be ignored, and the deployed system will not be added to a domain. If the Domain name field is specified, the User name and Password fields become mandatory. However, the Organizational unit field remains optional. If the Organizational unit field is not specified, the computer account will be stored in the default Computers container located under the Active Directory domain root.

Important: Windows computer names must be 15 characters or less in length and are derived from the corresponding host names in DNS. DNS host names, which are more than 15 characters in length, may cause duplicate computer names by keeping the first 15 characters of the DNS host names. In the case of a duplicate computer name, when the computer is joined to an Active Directory domain, it will either
replace the existing computer account or result in an error indicating that the account already exists. Since both are undesirable results, it is recommended that DNS host names be 15 characters or less in length. Provide the following domain information:

**Domain name**  
Specify the name of the domain to join, for example, cloud.company.com.

**User name**  
Specify the user name that is authorized to add a computer account to the domain. Refer to Microsoft documentation for details.

**Password**  
Specify the password of the domain user specified in User name.

**Organizational unit**  
Specify the organizational units where the computer account is stored. For example: ou=Computers,ou=ou1,dc=cloud,dc=company,dc=com where Computers and ou1 are the organizational units created under the Active Directory domain root, and cloud, company, and com are derived from the domain name cloud.company.com.

**Windows key management service**  
Provide the following KMS server information to be used for KMS client activation:

**KMS server IP address**  
Specify the IP address of the KMS server in your environment.

**KMS server port**  
Specify the port used for KMS service.

**Environment limits**  
The environment limits section enables you to provide the limits of the virtual CPU, virtual memory, and storage that this environment profile can use on the hypervisor. To provide virtual CPU, virtual memory, and storage limits, click the number in the limit column.

**Note:** IBM Cloud Orchestrator does not report information related to reserved resources and limits associated to the resource pool in VMware. By default, the resource pool is considered as unlimited. Use the environment profile in IBM Cloud Orchestrator to manage these resource limits.

**Access granted to...**  
Click this field to specify access to this environment profile for other users. Select users to make the environment profile readable or writable to these users. Initially this field is set to the role of the owner of the environment profile.

By default, the **Add more** box contains the **Everyone** built-in project. When a project has been added, click the link beside the entry to toggle between the following access levels:

- Read
- Write
- All
Click the link name of the project to show information about that project. You can also click the remove link to remove access for a project.

**Results**

When you have completed these steps, you have configured basic information about the environment profile.

**What to do next**

If there are no errors and all the resources the environment profile contains are operational, you can deploy it to the cloud or clouds you specified.

**Cloning an environment profile**

You can clone environment profiles that are created in IBM Cloud Orchestrator with the user interface. Cloning an environment profile provides a starting point for configuring a new environment profile as you can reuse some of the existing configuration.

**Before you begin**

Select an environment profile that most closely meets your needs, with the hypervisor type you want to use. The hypervisor type cannot be changed when you clone an environment profile.

If the profile is to deploy in a cloud other than the one specified in the profile you are cloning, have a cloud group configured and ready. All hypervisors must be configured and available in a cloud to create an environment profile that is ready to be deployed.

**About this task**

This task provides the necessary steps to clone an environment profile and then customize the copy to meet the needs of your environment.

**Procedure**

1. From the left panel of the Environment Profiles window, click the profile you want to clone. The description and general information about this environment profile display in the right panel of the Environment Profiles view.
2. Clone the environment profile. Click the clone icon on the upper right panel of the Environment Profiles view.
3. Provide the following basic information about the new environment profile you are cloning:
   - **Name** Enter a new unique name for the environment profile in the **Name** field. This information is required.
   - **Description** Optionally, enter a detailed description to identify and differentiate the environment profile in the **Description** field.
4. Click OK to save your changes. When the information is processed, you return to the Environment Profiles view and the profile you created is added to the list in the left panel. It is selected so that the information about it is shown in the right panel. For more information about the fields on this panel, see "Environment Profiles window fields" on page 351.
5. Edit the environment profile. You can edit the fields described in "Editing an environment profile."

Results

When you have completed these steps, you have cloned and customized the environment profile.

Editing an environment profile

You can edit some of the configuration for any environment profile to which you have access. You can modify environment profiles to suit the changing needs of your environment.

About this task

You can use environment profiles to track CPU, memory, storage and stop deployments at a particular size. If you have administrative permission, you can change the high water marks accordingly. Each profile can indicate how many resources of the cloud IBM Cloud Orchestrator can consume. This task provides information about the configuration that you can edit for existing environment profiles.

Procedure

1. From the left panel of the Environment Profiles window, select the environment profile to edit. The information about that environment profile is shown in the right panel of the Environment Profiles view.
2. Optional: Determine your access. If you are not able to edit the environment profile, check the Access granted to: field on the lower right panel to verify that you have access. If you do not have access, you can click the link on the owner, view the contact information, and contact the owner to ask for access.
3. Optional: Edit the following configuration information:
   a. Edit the description. Add or change the description of the environment profile in the Description field.
   b. Change the environment. Select a different environment, in which your environment profile is to run, in the Environment field. The following options are available:
      • All
      • Development
      • Test
      • Quality Assurance
      • Performance
      • Research
      • Production
      • Pre-Production
   c. Specify or change the format of the virtual machine name. In the Virtual machine name format field, you can specify the format for the virtual machine name, for example d_${hostname}.
   d. Specify how the IP addresses are provided. In the IP addresses provided by field, select one of the following options to specify how the IP addresses are provided:
      Pattern deployer
      If you choose to provide the IP address for a virtual machine at
deployment, then you must also specify the cloud group, IP group, host name, and IP address for each part.

**Important:** If you choose this option, then the person deploying the pattern cannot specify an IP address that is contained within the IP groups that are defined in IBM Cloud Orchestrator.

**IP Groups**
If IBM Cloud Orchestrator provides the IP address for a virtual machine, you only specify the cloud group and IP group for the pattern parts. IBM Cloud Orchestrator provides the IP address information.

d. Add, remove, or change the alias name for the cloud group in which the environment profile is to run.

**Add**
To add a cloud group, click the entry field under the **Deploy to cloud groups** label and select the cloud group to add.

**Remove**
Click the **Remove** link beside any listed cloud groups to remove them from the environment profile.

**Change alias name**
In the **Alias** field, change the name of the cloud. This name is shown at deployment.

e. Add, remove, or rename IP groups. Select or clear the **In use** box to indicate the IP groups in each cloud group to be used. You can also change the name of the IP group, as it is shown at deployment, in the **Alias** field.

f. Expand the **Windows domain information** field, to modify the domain information.

g. Expand the **Windows key management service** field, to modify the KMS server information.

h. In the **Environment limits** field, you can modify the limits of the virtual CPU, virtual memory, and storage.

i. Grant or remove access to the environment profile to projects. Use the **Access granted to** field to add, remove, or change access to this environment profile.

**Results**

If the hypervisors and resources for the cloud group specified are available, the environment profile can be deployed to the cloud group.

**Environment Profiles window fields**
The Environment Profiles window provides fields to group related deployment configuration options, like virtual machine names, IP address assignment, and cloud groups, together. With this configuration information grouped, you can manage the configuration in the environment profile without changing the pattern that is deploying the environment profile.

The following icons are on the upper right top bar of the Environment Profiles window:

**Refresh**
Refreshes the profile display in the configuration panel after any changes.

**Clone**
Clones the selected profile. You can clone the profile and then edit the copy.
Delete Deletes the profile from IBM Cloud Orchestrator.

There are two interactive panels of the Environment Profiles window:

Left panel
The left panel provides the following function:
- The **New** button to create new environment profiles
- A search function to locate existing environment profiles
- A list of environment profiles that have been created

Right panel
The right panel provides the fields to define and view details about a selected environment profile.

The fields on the right panel of the Environment Profiles window provide details about an environment profile selected from the listing on the left panel. The following fields define the selected environment profile:

**Description**
An editable field that provides the description of the profile. The description can be added or edited after the environment profile is created.

**Hypervisor type**
Shows **OpenStack** as the type of hypervisor with which the profile was created.

**Environment**
The environment is specified when the environment profile is created but it can be changed. The following environments can be specified:
- All
- Development
- Test
- Quality Assurance
- Performance
- Research
- Production
- Pre-Production

The default value is **All**.

**Created on**
Shows the time stamp when the profile was created.

**Current status**
Provides the status of the profile. This field shows if the environment profile is complete or if information is needed.

**The success icon**
The success icon indicates that the environment profile is complete and resources are available.

**The warning icon**
The warning icon indicates that environment profile is incomplete. A textual explanation, in addition to the warning icon, provides an explanation of the problem, or problems, with the environment profile configuration.

**Updated on**
Shows the timestamp of the most recent update.
Virtual machine name format
This optional field is a free form editing space to indicate the format of the virtual machine, for example, `d_${hostname}`. This field displays **None provided** initially.

IP addresses provided by
This field provides the following options:

**IP Groups**
Indicates that the IP address is to be provided by IBM Cloud Orchestrator at deployment. **IP Groups** is the default setting.

**Pattern deployer**
Indicates that the IP address is to be provided by the person deploying the pattern at the time of deployment.

**Important:** If this option is selected, the person deploying the pattern cannot specify an IP address that is contained within IP groups that are defined in IBM Cloud Orchestrator.

Deploy to cloud groups
Shows the following information for each cloud group in the list:

- **Name**  Shows the name of the IP group in the selected cloud.
- **Alias**  An entry field to specify an alias for the IP group for use in the environment profile. The default setting is the actual name of the IP group. Click to change the alias name.

**remove**
Removes the cloud group from the environment profile.

Clicking the expand icon shows the following additional fields for the selected cloud group:

**Using Environment profile**
Selection box to specify the IP group to use.

- **Name**  The name of the cloud group.

**Deploy to cloud groups**
The cloud groups to which this environment profile can deploy.

**Subnet address**
Shows the subnet address of the IP group.

- **Gateway**
  Shows the gateway address of the IP group.

- **Netmask**
  Shows the netmask address of the IP group.

Windows domain information
Shows the following domain information:

- **Domain name**
  Shows the name of the domain.

- **User name**
  Shows the user name that is authorized to add a computer account to the domain.

- **Password**
  Shows the password of the domain user specified in **User name**.
Organizational unit
   Shows the organizational units where the computer account is stored.

Windows key management service
   Shows the following KMS server information:

   KMS server IP address
   Shows the IP address of the KMS server in your environment.

   KMS server port
   Shows the port used for KMS service.

Environment limits
   In the table, you can set the following types of environment profile limits:
   - Virtual CPU
   - Virtual Memory
   - Storage
   This table also shows the current usage and the reserved usage for each of these types.

Access granted to
   By default, the user who created the environment profile has access to it and other users cannot edit it. This field can be edited and to provide access to this environment profile for projects. Selecting projects makes the environment profile readable or writable to the users belonging to these projects.

   By default, the Add more box contains the Everyone built-in project. When a project has been added, click the link beside the entry to toggle between the following access levels:
   - Read
   - Write
   - All
   Click the link name of the project to show information about that project. You can also click the remove link to remove access for a project.

Comments
   A comments field is provided to enable administrators to communicate information with one another regarding environment profiles.

Managing script packages
   You can use script packages to customize the behavior of parts in IBM Cloud Orchestrator topologies by adding script packages to pattern topologies. You can create script packages and then add them to the part you want to modify within the pattern containing that part.

Before you begin
   You must create a compressed file in .zip or .tgz (.tar.gz) format that contains the main executable file and all associated artifacts that support the execution of the main executable file. This will be uploaded into IBM Cloud Orchestrator and used as input to create the script package. See “Script packages overview” on page 355 for more information about using script packages with IBM Cloud Orchestrator.
The compressed file includes the script file (script.sh on Linux, or script.bat or script.cmd on Windows) in addition to the .json file needed to run the script on the deployed virtual machine. For more information, see “Configuring a script package using a JSON object” on page 364.

Script packages overview

Script packages can be added to pattern topologies to customize the behavior of the parts. Patterns are used to define cells and they can include script packages to further define behavior.

Script packages are simple containers that contain all the required artifacts necessary to run a script. The script package is a directory compressed into a single file that is uploaded to the catalog and then associated with patterns. The code included in the script package can be as simple as a .war file or as complex as a complete product. The content of a script package is not defined by IBM Cloud Orchestrator. The script included in the script package defines the required content for that package.

During deployment, script packages are transferred to the target virtual machines at a location you specify in the configuration. After they have transferred, they are extracted in that same location. When the virtual machines successfully start and are federated (if applicable), script packages are extracted. The scripts are run using the supplied command line. These files are written to the file system as the root user. If a different user requires access to these files, ensure that you set the correct user properties for the files on the guest operating system.

By default, your scripts are run on the deployed virtual machines using the root user context (on Linux) or administrator (on Windows). You can assume that a different user context for running the script package should contain a shell script that performs the following command: su virtuser -c "./nextShellScript.sh". This command starts a second script that runs in the user context of the virtuser. By running script packages using this method, you ensure that new files, directories, and binary files are created with the appropriate user context.

By default, the application server is started before scripts are run. To run a script, before starting the application server, use the AUTOSTART environment variable. By default, the AUTOSTART value is set to TRUE (AUTOSTART=TRUE) and the servers are started before the scripts are run. You can change the default value to FALSE (AUTOSTART=FALSE) to run a script before the server is started. For more information about environment variables, see “Script package environment variables” on page 365.

Scripts run in a prescribed order. If you are running an IBM WebSphere Application Server Hypervisor Edition script on multiple virtual machines, for example, then the script runs on the virtual machines in the following order:
1. Stand-alone Nodes
2. Deployment Manager
3. Job Manager (version 7.0.0.x patterns only)
4. Administrative Agent (version 7.0.0.x patterns only)
5. Custom Nodes
6. IBM HTTP Server
7. On-demand router (if you are using WebSphere Application Server 7.0.0.17 with Intelligent Management Pack)
If multiple script packages are included with a pattern, by default, the scripts are run in the same order they were added to that pattern. You can change the order in which script parts are run in the pattern. For more information, see “Ordering parts to run at deployment” on page 422.

When scripts are run by IBM Cloud Orchestrator, the IBM Cloud Orchestrator establishes a run time environment on the virtual machine using a Secure Shell (SSH) tunnel (this is valid only on Linux). By default, on the included Linux based virtual machines, this is the bash directory. This includes the definition of a set of environment variables. For more information about these environment variables, see “Script package environment variables” on page 365.

Script packages remain on the virtual machine after they are run. As they exist on the virtual machine, you can manually run your scripts from the virtual machine after deployment. To set the environment variables to run a script manually on the virtual machine, you must source the /etc/virtualimage.properties file (this is valid only on Linux). If you want the script package to be removed after it has run, you can build the script to delete itself.

In addition to the included scripts, you can review the examples that are provided in the subsection. The examples demonstrate how to use scripts in a virtual environment.

### Using script packages with the user interface

You can use the user interface to manage script packages. You can later add script packages to parts in system topologies to customize the behavior of virtual system patterns.

#### Before you begin

You must create a compressed file in .zip or .tgz (.tar.gz) format that contains the main executable file and all associated artifacts that support the execution of the main executable file. This will be uploaded into IBM Cloud Orchestrator and used as input to create the script package. See “Script packages overview” on page 355 for more information about using script packages with IBM Cloud Orchestrator.

The compressed file includes the script file (script.sh on Linux, or script.bat or script.cmd on Windows) in addition to the .json file needed to run the script on the deployed virtual machine. For more information, see “Configuring a script package using a JSON object” on page 364.

#### About this task

You can use the user interface to work with script packages.

#### Procedure

1. Navigate to the Script Packages window by clicking PATTERNS > Pattern Design > Script Packages from the menu.
2. Perform any of the following tasks:
   - Add a script package. For information about adding a script package, see “Adding a script package” on page 357.
   - Making a script package read-only. See “Making script packages read-only” on page 360 for more information.
• Associate a script package with a pattern. After adding the script package, you can associate it with a pattern. For information about associating a script package with a pattern, see "Associating a script package with a pattern" on page 361.

• Delete a script package. If you determine that a script package is no longer needed, you can delete it. For information about deleting a script package, see "Deleting a script package" on page 363.

Results

When the script package is created and associated with a specific pattern, the script package runs when the pattern is deployed. If you delete a script package, it is no longer available for use and this operation cannot be undone.

Adding a script package
You can create a script package and associate the script package with a pattern and defined cell. The script package is added to the catalog through the user interface.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

You must create a compressed file in .zip or .tgz (.tar.gz) format that contains the main executable file and all associated artifacts that support the execution of the main executable file. This will be uploaded into IBM Cloud Orchestrator and used as input to create the script package.

Important: When creating your scripts, ensure that you use a text editor that does not introduce control characters as this causes the scripts to possibly become unusable.

About this task

Use this task to add a new script package, that you have created, to a part within a pattern to manipulate the behavior of that part.

Procedure

1. From the upper left of the Script Packages window, click Create New. Additionally, you can clone an existing script package to create a copy of that script package. The cloned script package can then be modified to your specification. For more information about cloning a script package, see "Cloning a script package" on page 360.

2. Select the script package archive and click Import. The new script package displays in the left panel of the Script Packages window. The right panel displays configuration options for the script package.

3. Configure the script package.

The script package can be configured manually or by including a special JSON object in the script package. See "Configuring a script package using a JSON object" on page 364 for more information about configuring a script package using a JSON object.

The following information is required to configure a script package:

Script package files

Specifies the name and location of the compressed file that contains the
script package. Locate the local file system and select this file. After selecting the file, click **Upload** to copy the file to IBM Cloud Orchestrator. Only one file can be uploaded to a script package.

**Environment**

Defines a set of environment variables that are available when the script package is run on its target virtual system instance. The environment variables are a set of key/value pairs that are defined to the run time environment of the script. IBM Cloud Orchestrator supplies a set of environment entries for you.

See “[Script package environment variables” on page 365](#) for a listing of available environment variables.

In this section, you can also specify additional values that are specific to your deployment. The environment variable is added as a parameter in the pattern. The value for this environment variable is then provided when you, or another user you have provided with access to the pattern, deploys the pattern. A default value can be specified in the pattern.

**Working directory**

Specifies the location on the target virtual machine that IBM Cloud Orchestrator extracts the script package. The working directory is also the initial current working directory when the script is run.

**Logging directory**

Specifies the location of the logs generated by the script after it has been run. These logs can be accessed for viewing from either IBM Cloud Orchestrator or by directly accessing them on the virtual machine.

**Executable**

Specifies the command to be started for this script package. This can be an executable command already on the virtual machine, for example `wsadmin`, `tar`, `ant`, or another system command. You can also provide your own script to be run as part of the script package.

**Arguments**

Specify the command line that is passed to the executable command. This field can optionally contain environment variables and other valid command-line syntax. You can also access environment using standard techniques (for example `shell`, or `ant`).

**Timeout**

Specifies the maximum amount of time (in milliseconds) to wait for this package to finish running on the virtual machine. The default value is 6000000 (16 hours and 40 minutes).

**Executes**

Specifies when the script package is run on the product machine. The default behavior is for the script package to run after all the virtual machines have successfully started and all the nodes are federated, where applicable. The **default behavior occurs when the virtual system instance is created**. See “[Script packages overview” on page 355](#) for more information about when script packages are run.

**at virtual system instance creation**

Specifies that the script is run when the virtual system has finished starting during the initial creation.
at virtual system instance creation and when I initiate it
   Specifies that the script is run when the virtual system has
   finished starting during the initial creation and it can be started
   manually using the start icon that is displayed next to the script
   name for a virtual machine.

at virtual system instance deletion
   Specifies that the script is run when the virtual system is
   deleted.

   Important: Scripts run at virtual system instance deletion are
   only run if the virtual system instance is running when it is
   deleted.

when I initiate it
   Specifies that the script is started manually using the start icon
   that is displayed next to the script name for a virtual machine.
   Click the icon to run the script. There is no limit on the number
   of times a script is run using this method.

Included in patterns
   When you have added the script package to a pattern, a list of the
   patterns to which the script package belongs are displayed in this field.
   Each pattern name is a link that can be clicked to view the pattern. This
   field is initially blank.

   See “Working with virtual system patterns (classic)” on page 411 for
   more information about patterns.

In the cloud now
   Specifies a list of the cloud instances to which the script package
   belongs.

Access granted to
   Specifies the users and projects assigned access to this script. Scripts
   can be added and used by patterns that are created by these users or
   by the users belonging to these projects. Access is automatically granted
   to the user that creates the script package. If additional users require
   access to the script package, you must manually add the projects to
   which these users belong.

4. When you have completed the configuration for the script package, the script
   package is saved. You can exit from the Script Packages view.

Results

The script package is created and any users with access can use it with patterns.

What to do next

You can now associate this script package with a pattern. For more information
about patterns, see “Working with virtual system patterns (classic)” on page 411.
Cloning a script package
You can create a script package based on an existing script package. The cloned script package can then be modified to your specifications.

Before you begin
You must be assigned the catalogeditor role or the admin role to perform these steps.

Important: When creating scripts, ensure that you use a text editor that does not introduce control characters. Control characters can cause the scripts to become unusable.

About this task
Use this task to add a clone an existing script package.

Procedure
1. From the left panel of the Script Packages window, locate the script package to clone. If the script package that you want to clone is not displayed, you can search for the script package. To search for the script package, on the left panel of the Script Packages view, enter all or part of the name of the script package in the search field.
2. Select the script package to clone. Click the script package to clone from the list on the left panel of the Script Packages view. Details about the script package are displayed in the right panel of the Script Packages view.
3. Click the clone icon to clone the script package you have selected.
4. Enter the name for the cloned script package and click OK. The details panel for the new script package displays and can be modified.
5. When you have completed the configuration for the script package, the script package is saved when you exit the Script Packages view.

Results
After you have completed these steps, users or groups with access, can use the script package with patterns.

What to do next
You can now associate this script package with a pattern. See "Working with virtual system patterns (classic)" on page 411 for more information about patterns.

Making script packages read-only
Either draft or read-only script packages can be deployed for testing or production, but making a script package read-only prevents further edits. Making script packages read-only provides consistent reuse in the cloud.

Before you begin
You must be assigned the catalogeditor role or the admin role to perform these steps.
About this task

You can make a script package read-only to prevent further editing to the script package.

Procedure

1. Select the script package. From the left panel of the Script Packages window, select the script package. Script packages that have the read-only symbol by them are already read-only and cannot be edited. Script packages with the edit symbol beside them are not read-only and can be edited. Basic information about the selected script package is shown in the right panel of the Script Packages window.
2. Made the script package read-only to lock it to future editing. Click the Lock icon in the upper right toolbar of the Script Packages window.
3. Verify that you want to make the script package read-only. When prompted to verify that you want to make the script package read-only, click OK to lock the script package.

Results

When you have made the script package read-only.

What to do next

You can now associate the script package with a pattern. See “Working with virtual system patterns (classic)” on page 411 for more information about patterns.

Associating a script package with a pattern

You can associate a script package with a pattern and defined cell through the user interface.

Before you begin

You must create and configure the script package before you can associate it with a pattern. For information about creating and configuring a script package, see “Adding a script package” on page 357.

About this task

Use this task to associate a script package with a pattern.

Procedure

1. Navigate to the Virtual System Patterns window by clicking PATTERNS > Pattern Design > Virtual System Patterns for Virtual Systems, or by clicking PATTERNS > Pattern Design > Virtual System Patterns (Classic) for Virtual Systems (classic).
2. Select a pattern. In the left panel of the Virtual System Patterns window, select the pattern with which to associate the script package. The pattern must not be read-only. For more information about patterns and editing them, see the “Virtual system pattern (classic) editing views and parts” on page 419 topic. Basic information about the selected pattern displays in the right panel of the Virtual System Patterns view.
3. Edit the pattern. Click **Edit** on the upper right of the Virtual System Patterns (classic) window for virtual systems (classic) or click **Open** on the upper right of the Virtual System Patterns window for virtual systems (classic) to edit the pattern.

4. Select **Scripts**. From the drop-down box in the left panel of the Pattern Editor, for virtual systems (classic), or in the left panel of the Pattern Builder, for virtual systems, click **Scripts**. A list of the script package parts is provided that can be dropped into the virtual image parts on the right panel of the Pattern Editor view. This list can contain any script packages that you have provided for use with IBM Cloud Orchestrator. Script packages can then be added to the virtual image parts.

5. Add a script package. Any script packages you have defined to IBM Cloud Orchestrator are available in the list of script packages on the left panel of the Virtual System Patterns view. You can drop any script package from this list onto the virtual image parts on the canvas on the right. This associates the script package with that part.

   If a script runs on multiple virtual machines on the pattern, then the script runs on the virtual machines in the following order:
   a. Stand-alone Nodes
   b. Deployment Manager
   c. Job Manager (version 7.0.0.x patterns only)
   d. Administrative Agent (version 7.0.0.x patterns only)
   e. Custom Nodes
   f. IBM HTTP Server

   If multiple script packages are included with a pattern, then the scripts are run in the same order they were added to that pattern.

6. Optional: Configure any properties defined in the script package. Properties added to script packages can be defined when associating the script package with a part or it can be defined during deployment. Click the edit properties icon to set the value now. It is possible to use a variable syntax to set the value for properties where the value is not yet known. For more information about setting the value of a property to be variable, see "Properties variable syntax" on page 367.

**Results**

You have added one or more script packages to the virtual images on the pattern.

**What to do next**

When you have associated the script package with a pattern, you can complete configuration of the pattern and deploy it to the cloud group.
Deleting a script package
You can manually delete a script package, disassociating it with the pattern and cell.

Before you begin
You must be assigned the catalogeditor role and be granted all access to the script package you want to remove. You can also perform these steps if you are assigned the admin role.

Script packages that are read-only, or that are in use, cannot be deleted. Before you delete a script package, ensure that it is no longer needed by the pattern and the cell with which it is associated.

About this task
This task provides information about manually deleting a script package from the Script Package panel of the IBM Cloud Orchestrator user interface.

Procedure
1. From the left panel of the Script Packages window, locate the script package. If the script package that you want to delete is not displayed, you can search for the script package. To search for the script package, from the left panel of the Script Packages view, enter all or part of the name of the script package in the search field.
2. Select the script package to delete. Click the script package to delete from the list on the left panel of the Script Packages view. Details about the script package are displayed in the right panel of the Script Packages view.
3. Determine if the script package can be deleted. A script package can only be deleted if it is not:
   - Marked as read-only. The read-only icon is displayed in the listing of script packages if it is read-only.
   - Included in any patterns. If it is included in any patterns, the delete icon is not available and the Included in patterns field displays the linked patterns for which this script package is included.

   If the script package is referenced by any patterns, you can click the pattern name link in the Included in patterns field to go to the Virtual System Patterns panel for that pattern. From this panel, you can remove the script package from the pattern.
4. Delete the package. Click the delete icon on the upper right of the Script Packages view.
5. Confirm the deletion. You are prompted to confirm that you want to delete the selected script package. Click OK to delete the script package.

Results
The script package is deleted from IBM Cloud Orchestrator.
Configuring a script package using a JSON object

A special JSON object, `cbscript.json` can be used to populate all the information needed to configure a script package when it is added to the catalog.

Overview

When you add a new script package to the catalog, you can include the configuration information in the compressed file (zip files and .tgz files are supported) that is uploaded using a special JSON object. The JSON object must be named `cbscript.json` and be included at the root of the compressed file that is uploaded. Including a `cbscript.json` file in your script package is useful when sharing scripts. An additional benefit is gained in limiting typographical errors as the information only needed to be entered once because the default values are pre-populated with the information in the `cbscript.json` file.

Note: You must provide a formatted JSON content in the `cbscript.json` file in addition to following the JSON specification. For example, you must make sure it always puts a carriage return after a JSON attribute:value entry, and avoid multiple quoted sets on a single line.

Example of `cbscript.json` file

```json
[
    {
        "name": "MyScriptPackage",
        "version": "1.0",
        "description": "This script package installs the specified application",
        "command": "/bin/sh ${WAS_PROFILE_ROOT}/bin/wsadmin.sh",
        "log": "${WAS_PROFILE_ROOT}/logs/wsadmin.traceout",
        "location": "/opt/tmp/installapp",
        "timeout": "0",
        "ostype": "linux/unix",
        "commandargs": "-lang jython -f /opt/tmp/installapp/install_app.jy $APP_LOCATION $INSTALL_ARGS",
        "keys": [
            {
                "scriptkey": "APP_LOCATION",
                "scriptvalue": "",
                "scriptdefaultvalue": ""
            },
            {
                "scriptkey": "INSTALL_ARGS",
                "scriptvalue": "",
                "scriptdefaultvalue": ""
            }
        ]
    }
]
```
Script package environment variables

When you are defining a script package to IBM Cloud Orchestrator, there are a set of supplied environment variables that can be used.

Purpose of environment variables

You can use the environment variables provided by IBM Cloud Orchestrator for defining script packages. Environment variables are provided through the virtual image and after deployment are located on each virtual machine at /etc/virtualimage.properties (on Linux) and at \windows\setup\ibm\virtualimage.properties on Windows. The set of environment variables that are available is determined by what environment variables exist on the virtual machine.

Specify the environment variables in the Environment field of the Script Packages window. For more information, see “Adding a script package” on page 357.

You can use environment variables to define your script packages for use with pattern topologies. For more information about pattern topologies, see “Working with virtual system patterns (classic)” on page 411.

Life cycle variables

A set of life cycle variables is included with most virtual images. These environment variables specify many of the scripts and locations required to manage the life cycle of the virtual system instances.

Install service

SERVICE_COMMAND_LOCATION="/var/adm/virtualimages/bin"

Install service

SERVICE_COMMAND="installService.sh"

Install service

SERVICE_PACKAGE_LOCATION="/tmp/update"

Install service

OS_SERVICE_PACKAGE_LOCATION="/tmp/update/os"

Install service

APP_SERVICE_PACKAGE_LOCATION="/tmp/update/app"

Reset virtual image

RESET_VIRTUAL_IMAGE_COMMAND_LOCATION="/var/adm/ibmvmmcoc-postinstall"

Reset virtual image

RESET_VIRTUAL_IMAGE_COMMAND="resetvm.sh"

Start virtual image services

START_SERVICES_COMMAND_LOCATION="/var/adm/virtualimages/bin"

Start virtual image services

START_SERVICES_COMMAND="startVirtualImageService.sh"

Stop virtual image services

STOP_SERVICES_COMMAND_LOCATION="/var/adm/virtualimages/bin"
Stop virtual image services
STOP_SERVICES_COMMAND="stopVirtualImageService.sh"

WebSphere Application Server variables
The following variables exist and can be used for IBM WebSphere Application Server Hypervisor Edition virtual images.

Number of Profiles to create (AdminAgent)
PROFILE_NUMBER=

Autostart WebSphere Servers
AUTOSTART=true

Federate Node (true | false)
DMGR_FEDERATE=false

Register with Job Manager (true | false)
JMGR_REGISTER=false

WebSphere Application Server operation commands to use for start and stop services commands
OPERATION_COMMAND_LOCATION="/opt/IBM/AE/AS"

WebSphere Application Server operation commands to use for start and stop services commands
OPERATION_COMMAND="${WAS_PROFILE_ROOT}/bin/ws_ant.sh -f /opt/IBM/AE/AS/wasHVControl.ant"

WebSphere Administrative Console URL
ADMIN_CONSOLE_URL=

WebSphere Cell Name
CELL_NAME=RainmakerCell0

WebSphere Default Profile location
WAS_PROFILE_ROOT=/opt/IBM/WebSphere/Profiles/
DefaultAppSrv01

WebSphere Install Root
WAS_INSTALL_ROOT=/opt/IBM/WebSphere/AppServer

WebSphere Install Root
PROFILE_ROOT=/opt/IBM/WebSphere/Profiles

WebSphere Install Root
HOSTNAME=vm-009-097.rainmaker.raleigh.ibm.com

WebSphere Node Name
NODE_NAME=RainmakerNode0

WebSphere Profile Name
PROFILE_NAME=DefaultAppSrv01

WebSphere Profile Type
PROFILE_TYPE=default

These two variables are dynamic. IBM Cloud Orchestrator adds them each time a script requiring them is run. After the script has completed, these environment variables are removed.

WebSphere Administrative Password
WAS_PASSWORD=password

WebSphere Administrative Username
WAS_USERNAME=virtuser
Related concepts:
“Script packages overview” on page 355

Script packages can be added to pattern topologies to customize the behavior of the parts. Patterns are used to define cells and they can include script packages to further define behavior.

Related tasks:
“Associating a script package with a pattern” on page 361
You can associate a script package with a pattern and defined cell through the user interface.

Properties variable syntax
A properties variable syntax, ${...}, is used for variables in script packages that are not known until deployment time.

Certain values, such as host names, are not always known before deployment. In IBM Cloud Orchestrator you can still reference these properties in your script package by using a certain syntax. This format can be used when associating the script with a part or it can be used when the pattern is deployed.

The general syntax for these variables is ${part-name_.property-name}. Each part has a unique name that is not translated that can be used like a variable. You can find this unique part name by clicking the properties icon of the part. The value in the Name field is the value that can be used as the part-name. When a part name contains spaces, such as Core 05, the syntax does support the use of the space character.

The property name can be any properties you have added to your script package. For example, if you have a part named DMGRPart and have an associated script that has a property called MYPROPERTY, you can use ${DMGRPart_.MYPROPERTY} to represent that future value of that property. In addition to custom properties, the property name can also be any of the following built-in values:

**Network-oriented variables**
- hostname
- domain
- ipaddr
- netmask
- gateway
- pri_dns
- sec_dns

**Note:** If you want to set the hostname variable for the Windows system, consider the limitation described in “Setting the host name when deploying a Windows system” on page 824.

**Locale-oriented variables**
- language
- country
- encoding

**WebSphere Application Server-oriented variables**
- cell_name
- node_name
Related tasks:

“Associating a script package with a pattern” on page 361
You can associate a script package with a pattern and defined cell through the user interface.

“Working with virtual system patterns (classic)” on page 411
Using a virtual system pattern, you can describe the topology of a system that you want to deploy. Virtual system patterns provide repeatable system deployment that can be reproduced. To build virtual system patterns, you can use parts from one or more virtual images, add-ons, and script packages.

Script package examples
A number of example script packages are provided with the system. Additional examples of typical script packages are described.

Script packages provided with the system
The following script packages are included with the system in the catalog:

• Add IBM HTTP Server node
• Disable Root Login
• Vertical CPU Scaling
• Vertical Memory Scaling

In addition to these script packages, there are several sample compressed archive files provided with the Plug-in Development Kit that you can use as examples for creating your own script packages to provide vertical scaling of CPU and memory resources for middleware.

Add IBM HTTP Server node
The Add IBM HTTP Server node script package is a special script package that contains no code and no executable script. It contains only a single environment variable, \texttt{ihs = true}. This script package is valid for use only with classic virtual system patterns.

You can add this script package only to deployment manager parts or standalone server parts. The configuration of the IBM HTTP Server is part of the provisioning of the deployment manager or standalone server parts. Adding this script package indicates to these two parts that the IBM HTTP Server should be configured. When you add this script package to a deployment manager part, for example, you will see one web server in the Web Servers list (with Web Server Type=IBM HTTP Server).

If you add this script package to a different part type, for example, a custom node or an on demand router node, no action is taken. In these cases, IBM HTTP Server configuration is not processed, because if, for example, more than one custom node is deployed, there would be a IBM HTTP Server on every custom node. This is not the intent of the pattern.
Disable Root Login

You can add this script package to a pattern deploying a virtual Linux system. The script disables SSH login by the root user. You can also modify the script package to run on demand, and add this script package at a later time to disable the root login as needed.

Note: IBM Cloud Orchestrator must be able to log in to the virtual machine as root to run script packages and perform other management functions. IBM Cloud Orchestrator automatically generates a public/private key pair during deployment; it then uses the private key to log in to the virtual machine as root to run script packages. This script modifies the operating system configuration to allow remote root login only with this predefined private key.

When this script package is deployed with the image, it makes changes to the operating system configuration. For more information about these changes, see the related links.

The archive file for this script package is named disableRootLogin.zip. It contains the following files:

cbscript.json
This file contains the following parameter information:

```
[  
   {  
      "command": "\"/tmp/disableRootLogin/disableRootLogin.sh\"",  
      "description": "Disables Root Login for AIX and Linux",  
      "execmode": 0,  
      "filename": "disableRootLogin.zip",  
      "location": "\"/tmp/disableRootLogin\"",  
      "log": "\"/tmp/disableRootLogin/logs\"",  
      "name": "Disable Root Login",  
      "ostype": "linux/unix",  
      "timeout": 0,  
      "type": "APPLICATION"  
   }  
]
```

To modify the script package to run manually, change the value of execmode to 2.

extendedattributes.json
This file contains the following parameter information:

```
[  
   {  
      "savevars": 0  
   }  
]
```

The savevars parameter controls whether environment variables are saved after post-deployment executions. A value of 0 indicates that variables are not saved.

disableRootLogin.sh
This is the script that is called when the script package is run.

README.TXT
Some basic information is included in this file.
**Vertical CPU Scaling**

This is a generic scaling policy that uses operating system metrics to scale the number of vCPUs that are allocated to a virtual machine. You can add this script package to a pattern deploying a virtual Linux system.

You can use this as a starting point to define your own scaling policy, and you can add the resulting script package to any supported virtual system part. When this scaling policy is deployed, a monitoring and auto-scaling agent runs in each virtual machine to trigger scaling independent of other nodes.

The archive file for this script package is named `VerticalScaleGenericCPU.zip`. It contains the following files:

**cbscript.json**

This file contains the following parameter information:

```json

[  
  {  
    "command": "/tmp/VerticalScaleGenericCPU/enable_as.sh",  
    "description": "A generic scaling policy that uses operating system metrics to scale the number of vCPUs allocated to a virtual machine.",  
    "execmode": 0,  
    "filename": "VerticalScaleGenericCPU.zip",  
    "keys": [  
      {  
        "locked": false,  
        "required": true,  
        "scriptdefaultvalue": "1",  
        "scriptkey": "MIN_CPU_COUNT"  
      },  
      {  
        "locked": false,  
        "required": true,  
        "scriptdefaultvalue": "8",  
        "scriptkey": "MAX_CPU_COUNT"  
      },  
      {  
        "locked": false,  
        "required": true,  
        "scriptdefaultvalue": "80",  
        "scriptkey": "SCALE_UP_CPU_THRESHOLD"  
      },  
      {  
        "locked": false,  
        "required": true,  
        "scriptdefaultvalue": "0",  
        "scriptkey": "SCALE_DOWN_CPU_THRESHOLD"  
      },  
      {  
        "locked": false,  
        "required": true,  
        "scriptdefaultvalue": "300",  
        "scriptkey": "TRIGGER_TIME"  
      }  
    ],  
    "location": "/tmp/VerticalScaleGenericCPU",  
    "log": "/tmp/VerticalScaleGenericCPU",  
    "name": "VerticalScaleGenericCPU",  
    "ostype": "linux/unix",  
    "timeout": 0,  
    "type": "APPLICATION"  
  }
]
```

IBM Cloud Orchestrator 2.4: User's Guide
The following environment variables are defined:

**MIN_CPU_COUNT**
This is a required variable that indicates the minimum number of vCPUs that the virtual machine must have. The default value is 1.

**MAX_CPU_COUNT**
This is a required variable that indicates the maximum number of vCPUs to which the virtual machine can be scaled. The default value is 8.

**SCALE_UP_CPU_THRESHOLD**
This is a required variable that indicates the percent threshold of CPU usage that, when exceeded, triggers a scale up operation to be automatically performed on the virtual machine. This threshold must be exceeded for the duration of time indicated by the **TRIGGER_TIME** variable before scaling is performed. The default value is 80 percent.

**SCALE_DOWN_CPU_THRESHOLD**
This variable is set to 0 percent by default. Scaling down is not supported for IBM Cloud Orchestrator.

**TRIGGER_TIME**
This is a required variable that indicates the time, in seconds, that a threshold must be exceeded before a scaling operation is performed. The default value is 300 seconds. For example, if the CPU usage on a virtual machine exceeds the **SCALE_UP_CPU_THRESHOLD** value (80%) for 300 seconds (5 minutes), then a scale up operation is triggered to increase the vCPU count.

**extendedattributes.json**
This file contains the following parameter information:

```json
[
  {
    "savevars": "0"
  }
]
```

The **savevars** parameter controls whether environment variables are saved after post-deployment executions. A value of 0 indicates that variables are not saved.

**enable_as.sh**
This is the script that is called when the script package is run.

When you deploy a virtual system pattern that uses this script package, the deployment history (over time) shows when the number of CPUs is changed. A message is displayed, similar to the following example:

```
CPU changed for virtual machine auslpa165-OS Node-scalingCPUTest-766 from 2 cores to 3 cores
```

For each vertical scaling operation, the number of vCPUs is increased by one CPU.

**Vertical Memory Scaling**

This is a generic scaling policy that uses operating system metrics to scale the amount of memory that is allocated to a virtual machine. You can add this script package to a pattern deploying a virtual Linux system.
You can use this as a starting point to define your own scaling policy, and you can add the resulting script package to any supported virtual system part. When this scaling policy is deployed, a monitoring and auto-scaling agent runs in each virtual machine to trigger scaling independent of other nodes.

The archive file for this script package is named `VerticalScaleGenericMemory.zip`. It contains the following files:

**cbscript.json**

This file contains the following parameter information:

```json
[

{
  "command": "/tmp/VerticalScaleGenericMemory\enable_as.sh",
  "description": "A generic scaling policy that uses operating system metrics to scale the amount of memory allocated to a virtual machine.",
  "execmode": 0,
  "filename": "VerticalScaleGenericMemory.zip",
  "keys": [
  {
    "locked": false,
    "required": true,
    "scriptdefaultvalue": "4096",
    "scriptkey": "MIN_MEMORY"
  },
  {
    "locked": false,
    "required": true,
    "scriptdefaultvalue": "8192",
    "scriptkey": "MAX_MEMORY"
  },
  {
    "locked": false,
    "required": true,
    "scriptdefaultvalue": "80",
    "scriptkey": "SCALE_UP_MEMORY_THRESHOLD"
  },
  {
    "locked": false,
    "required": true,
    "scriptdefaultvalue": "0",
    "scriptkey": "SCALE_DOWN_MEMORY_THRESHOLD"
  },
  {
    "locked": false,
    "required": true,
    "scriptdefaultvalue": "300",
    "scriptkey": "TRIGGER_TIME"
  }
]

"location": "/tmp/VerticalScaleGenericMemory",
"log": "/tmp/VerticalScaleGenericMemory",
"name": "VerticalScaleGenericMemory",
"ostype": "linux\unix",
"timeout": 0,
"type": "APPLICATION"
}
]
```

The following environment variables are defined:

**MIN_MEMORY**

This is a required variable that indicates the minimum amount of memory that the virtual machine must have. The default value is 4096.
**MAX_MEMORY**
This is a required variable that indicates the maximum amount of memory to which the virtual machine can be scaled. The default value is 8192.

**SCALE_UP_MEMORY_THRESHOLD**
This is a required variable that indicates the percent threshold of memory usage that, when exceeded, triggers a scale up operation to be automatically performed on the virtual machine. This threshold must be exceeded for the duration of time indicated by the **TRIGGER_TIME** variable before scaling is performed. The default value is 80 percent.

**SCALE_DOWN_MEMORY_THRESHOLD**
This variable is set to 0 percent by default. Scaling down is not supported for IBM Cloud Orchestrator.

**TRIGGER_TIME**
This is a required variable that indicates the time, in seconds, that a threshold must be exceeded before a scaling operation is performed. The default value is 300 seconds. For example, if the memory usage on a virtual machine exceeds the **SCALE_UP_MEMORY_THRESHOLD** value (80%) for 300 seconds (5 minutes), then a scale up operation is triggered to increase the amount of memory allocated to the virtual machine.

**extendedattributes.json**
This file contains the following parameter information:

```json
[
  {
    "savevars": "0"
  }
]
```

The `savevars` parameter controls whether environment variables are saved after post-deployment executions. A value of 0 indicates that variables are not saved.

**enable_as.sh**
This is the script that is called when the script package is run.

When you deploy a virtual system pattern that uses this script package, the deployment history (over time) shows when the amount of memory is changed. A message is displayed, similar to the following example:

```
Memory changed for virtual machine auslps158-Standalone from 2048 to 3072
```

For each vertical scaling operation, the amount of memory is increased by 1024 MB.

**Vertical scaling of CPU for WebSphere Application Server Hypervisor Edition**

In addition to the generic script package that you can use for vertical scaling of CPU in your virtual machines, you can create your own script packages that you can use for vertical scaling of CPU for middleware components such as WebSphere Application Server, DB2, and others.

Typically if you need to tune your middleware configuration after a scaling operation to take advantage of the availability of additional resources, you might
need to provide your own content specific resource.py executable. The scaling function will locate and run this file to adjust middleware as needed based on the resource changes. For example, after performing a CPU scaling operation for WebSphere Application Server Hypervisor Edition, you might need to increase the thread pool size to take advantage of the additional vCPU resource.

A sample compressed archive file, WASHV_CPUBased.zip, is included in the Plug-in Development Kit, as an example for creating a vertical scaling script package for WebSphere Application Server Hypervisor Edition. See the related links for more information about downloading and unpacking the Plug-in Development Kit.

Use the standard procedure for creating a script package in the IBM Cloud Orchestrator catalog, and upload this archive file into the script package. The archive file contains the following files:

cbscript.json

This file contains the following parameter information:

```json
[

{
  "command": "/bin/sh /tmp/WASHV_CPUBased\enable_as.sh",
  "description": "A WASHV script example for scaling up based CPU",
  "execmode": 0,
  "filename": "WASHV_CPUBased.zip",
  "keys": [
    {
      "locked": false,
      "required": true,
      "scriptdefaultvalue": "1",
      "scriptkey": "MIN_CPU_COUNT",
      "scriptvalue": "1"
    },
    {
      "locked": false,
      "required": true,
      "scriptdefaultvalue": "8",
      "scriptkey": "MAX_CPU_COUNT",
      "scriptvalue": "8"
    },
    {
      "locked": false,
      "required": true,
      "scriptdefaultvalue": "80",
      "scriptkey": "SCALE_UP_CPU_THRESHOLD",
      "scriptvalue": "80"
    },
    {
      "locked": false,
      "required": true,
      "scriptdefaultvalue": "0",
      "scriptkey": "SCALE_DOWN_CPU_THRESHOLD",
      "scriptvalue": "0"
    },
    {
      "locked": false,
      "required": true,
      "scriptdefaultvalue": "120",
      "scriptkey": "TRIGGER_TIME",
      "scriptvalue": "120"
    }
  ],
  "location": "/tmp/WASHV_CPUBased",
  "log": "/tmp/WASHV_CPUBased",
  "name": "WAS_VerticalScalingCPUScriptPkg",
  "ostype": "linux/unix",
]```
The following environment variables are defined:

**MIN_CPU_COUNT**
This is a required variable that indicates the minimum number of vCPUs that the virtual machine must have. The default value is 1.

**MAX_CPU_COUNT**
This is a required variable that indicates the maximum number of vCPUs to which the virtual machine can be scaled. The default value is 8.

**SCALE_UP_CPU_THRESHOLD**
This is a required variable that indicates the percent threshold of CPU usage that, when exceeded, triggers a scale up operation to be automatically performed on the virtual machine. This threshold must be exceeded for the duration of time indicated by the **TRIGGER_TIME** variable before scaling is performed. The default value is 80 percent.

**SCALE_DOWN_CPU_THRESHOLD**
This variable is set to 0 percent by default. Scaling down is not supported for IBM Cloud Orchestrator.

**TRIGGER_TIME**
This is a required variable that indicates the time, in seconds, that a threshold must be exceeded before a scaling operation is performed. The default value is 120 seconds. For example, if the CPU usage on a virtual machine exceeds the **SCALE_UP_CPU_THRESHOLD** value (80%) for 120 seconds (2 minutes), then a scale up operation is triggered to increase the vCPU count.

This file contains the following parameter information:

```
[  
  {  
    "savevars": "0"  
  }  
]
```

The `savevars` parameter controls whether environment variables are saved after post-deployment executions. A value of 0 indicates that variables are not saved.

**enable_as.sh**
This is the script that is called when the script package is run.

**resource.py**
This is the script that is run automatically to perform the middleware tuning to take advantage of the change in CPU resource. In this example, the `scripts/was_web_thread.py` script is called, which adjusts the thread pool size for WebSphere Application Server.

The script contains the following code:

```python
import sys
import subprocess
import json
```
params = json.loads(sys.argv[1])

if int(params['newCpuCount']) > int(params['oldCpuCount']):
    cpu = params['newCpuCount']
    retcode = subprocess.call(['sh', '/opt/IBM/WebSphere/AppServer/bin/wsadmin.sh', '-lang', 'jython', '-f', '/tmp/WASHV_CPUBased/scripts/was_web_thread.py', str(cpu)])
    print 'wsadmin.sh return code:', retcode

was_webresource.py

This script is located in the scripts folder, and is called by resource.py to perform the middleware tuning to adjust the thread pool size for WebSphere Application Server.

The script contains the following code:

```python
import AdminUtilities

def getName(objectId):
    endIndex = (objectId.find("(c") - 1)
    startIndex = 0
    if (objectId.find("\") == 0):
        startIndex = 1
    return objectId[startIndex:endIndex+1]

assert len(sys.argv) == 1

get_cpu = int(sys.argv[0])

setMaxInt = 20 * target_cpu

setMaxStr = str(setMaxInt)

print "Set WebContainer's thread pool max size: %s, min size: %s %s" % (setMaxStr, setMaxStr)

try:
    theList = AdminControl.completeObjectName('WebSphere:*,type=ThreadPool,name=WebContainer')
    theList = theList.splitlines()
    for tp in theList:
        if tp.find('WebContainer') != -1:
            currMinSize = AdminControl.invoke(tp, 'getMinimumPoolSize')
            currMinSize = int(currMinSize)
            currMaxSize = AdminControl.invoke(tp, 'getMaximumPoolSize')
            currMaxSize = int(currMaxSize)
            if currMaxSize < setMaxInt:
                print "currMaxSize is %s" % currMaxSize
                print "new MaxSize is %s" % setMaxStr
                AdminControl.setAttribute(tp, [["maximumSize", setMaxInt], ["minimumSize", setMaxInt]])
except:
    print "WAS process is not running, only update the config file"

server1 = AdminConfig.getid('/Cell:CloudBurstCell_1/Node:CloudBurstNode_1/Server:server1/)
tpList = AdminConfig.list('ThreadPool', server1)
for pool in tpList.split("\n"):
    pool = pool.rstrip()
    if (getTitle(pool) == "WebContainer"):
        attrs = [["maximumSize", setMaxInt], ["minimumSize", setMaxInt]]
        AdminConfig.modify(pool, attrs)
        AdminConfig.save()
    break
```

Vertical scaling of memory for WebSphere Application Server Hypervisor Edition

In addition to the generic script package that you can use for vertical scaling of memory in your virtual machines, you can create your own script packages that you can use for vertical scaling of memory for middleware components such as WebSphere Application Server, DB2, and others.
Typically if you need to tune your middleware configuration after a scaling operation to take advantage of the availability of additional resources, you might need to provide your own content specific resource.py executable. The scaling function will locate and run this file to adjust middleware as needed based on the resource changes. For example, after performing a memory scaling operation for WebSphere Application Server Hypervisor Edition, you might need to adjust the JVM heap size to take advantage of the additional memory resource.

A sample compressed archive file, WASHV_MemoryBased.zip, is included in the Plug-in Development Kit, as an example for creating a vertical scaling script package for WebSphere Application Server Hypervisor Edition. See the related links for more information about downloading and unpacking the Plug-in Development Kit.

Use the standard procedure for creating a script package in the IBM Cloud Orchestrator catalog, and upload this archive file into the script package. The archive file contains the following files:

cbscript.json

This file contains the following parameter information:

```
[
   {
      "command": "\"/bin/sh /\tmp/WASHV_MemoryBased\(enable_as.sh\",
      "description": "A WASHV script example for scaling up based Memory",
      "execmode": 0,
      "filename": "WASHV_MemoryBased.zip",
      "keys": [
         {
            "locked": false,
            "required": true,
            "scriptdefaultvalue": "1024",
            "scriptkey": "MIN_MEMORY",
            "scriptvalue": "1024"
         },
         {
            "locked": false,
            "required": true,
            "scriptdefaultvalue": "4096",
            "scriptkey": "MAX_MEMORY",
            "scriptvalue": "4096"
         },
         {
            "locked": false,
            "required": true,
            "scriptdefaultvalue": "80",
            "scriptkey": "SCALE_UP_MEMORY_THRESHOLD",
            "scriptvalue": "80"
         },
         {
            "locked": false,
            "required": true,
            "scriptdefaultvalue": "0",
            "scriptkey": "SCALE_DOWN_MEMORY_THRESHOLD",
            "scriptvalue": "0"
         },
         {
            "locked": false,
            "required": true,
            "scriptdefaultvalue": "120",
            "scriptkey": "TRIGGER_TIME",
            "scriptvalue": "120"
         }
      ]
   }
]
```
The following environment variables are defined:

**MIN_MEMORY**
This is a required variable that indicates the minimum amount of memory that the virtual machine must have. The default value is 1024.

**MAX_MEMORY**
This is a required variable that indicates the maximum amount of memory to which the virtual machine can be scaled. The default value is 4096.

**SCALE_UP_MEMORY_THRESHOLD**
This is a required variable that indicates the percent threshold of memory usage that, when exceeded, triggers a scale up operation to be automatically performed on the virtual machine. This threshold must be exceeded for the duration of time indicated by the TRIGGER_TIME variable before scaling is performed. The default value is 80 percent.

**SCALE_DOWN_MEMORY_THRESHOLD**
This variable is set to 0 percent by default. Scaling down is not supported for IBM Cloud Orchestrator.

**TRIGGER_TIME**
This is a required variable that indicates the time, in seconds, that a threshold must be exceeded before a scaling operation is performed. The default value is 120 seconds. For example, if the memory usage on a virtual machine exceeds the SCALE_UP_MEMORY_THRESHOLD value (80%) for 120 seconds (2 minutes), then a scale up operation is triggered to increase the amount of memory allocated to the virtual machine.

**extendedattributes.json**
This file contains the following parameter information:

```
[
  {
    "savevars": "0"
  }
]
```

The `savevars` parameter controls whether environment variables are saved after post-deployment executions. A value of 0 indicates that variables are not saved.

**enable_as.sh**
This is the script that is called when the script package is run.

**resource.py**
This is the script that is run automatically to perform the middleware tuning to take advantage of the change in memory resource. In this example, the script calculates the adjustment needed to the heap size, then calls the `scripts/stopServer.py` script to stop the WebSphere Application
Server, then issues the command to start the server again. The WebSphere process must be restarted for the new JVM heap configuration to take effect.

The script contains the following code:

```python
import os
import sys
import json
import re
import shutil
import commands
import logging
import subprocess

def calculate_64bit_jvmmemory(matchobj):
    newHeapSize = int(matchobj.group(1)) + int(gap/1.5/128)*128
    if newHeapSize > 6144:
        return '6144'
    else:
        return str(newHeapSize)

def calculate_32bit_jvmmemory(matchobj):
    newHeapSize = int(matchobj.group(1)) + int(gap/1.5/128)*128
    if newHeapSize > 2048:
        return '2048'
    else:
        return str(newHeapSize)

def jvm_memory_str(matchobj):
    newHeapSize = int(int(parms['newMemory'])/1.5/128)*128
    if newHeapSize > 2048:
        newHeapSize = 2048
    strval = str(matchobj.group(1)) + ' initialHeapSize="128" maximumHeapSize="'
    return str(strval)

def startServer():
    #os.system('/opt/IBM/WebSphere/AppServer/bin/startServer.sh server1')
    subprocess.call(['sh', '/opt/IBM/WebSphere/AppServer/bin/startServer.sh', 'server1'])
    logger.debug("start Server...")

def stopServer():
    #os.system('/opt/IBM/WebSphere/AppServer/bin/stopServer.sh server1')
    command = '/opt/IBM/WebSphere/AppServer/bin/wsadmin.sh -conntype SOAP -lang jython -f /tmp/WASHV_MemoryBased/scripts/stopServer.py'
    subprocess.call(command, shell=True)
    logger.debug("stop Server...")

logger = logging.getLogger("resource.py")
parms = json.loads(sys.argv[1])

if int(parms['newMemory']) > int(parms['oldMemory']):
    gap = int(parms['newMemory']) - int(parms['oldMemory'])

    #originalFile =
    /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/config/cells/localhostNode01Cell/nodes/localhostNode01/servers/server1/server.xml'
originalFile =
    /opt/IBM/WebSphere/Profiles/DefaultAppSrv01/config/cells/CloudBurstCell_1/nodes/CloudBurstNode_1/servers/server1/server.xml'
backupFile = originalFile + '.bk'
    output = commands.getoutput('/opt/IBM/WebSphere/AppServer/bin/versionInfo.sh')
    #print output
if os.path.exists(originalFile):
    if not os.path.exists(backupFile):  
        shutil.copy(originalFile, backupFile)
    with open(originalFile) as cfgFile:
        data = cfgFile.read()
    m = re.search('32 bit',output)
    n = re.search('64 bit',output)
    if m:
        print 'IBM 32-bit SDK for Java'
    if t:
        print 'IBM 64-bit SDK for Java'
```

Chapter 7. Managing and deploying virtual patterns 379
if re.search('maximumHeapSize="2048"',data):
    print 'maximumHeapSize="2048" There is no extra memory size to scaling up'
    logger.debug("There is no extra memory size to scaling up")
    new_data = data
else :
    new_data = re.sub(r'(?<=maximumHeapSize=)\d+', calculate_32bit_jvmmemory, data)
else :
    new_data = re.sub(r'(?<=verboseModeJNI=)\d+', jvm_memory_str, data)

else :
    print 'Nothing changed...'
    logger.debug("There is no related information ")
if cmp(data,new_data) != 0 :
    with open(originalFile, "w") as cfgFile :
        cfgFile.write(new_data)
        Info = commands.getoutput("ps -ef | grep java")
        res = re.search('server1',Info)
        if res :
            print 'stopServer ....'
            stopServer()
        print 'startServer ....'
        startServer()
        else :
            print 'startServer ....'
            startServer()
    else :
        logger.debug("There is no configuration file to update ")

stopServer.py

This script is located in the scripts folder, and is called by resource.py to stop the WebSphere Application Server instance.

The script contains the following code:
AdminControl.stopServer("server1", "CloudBurstNode_1")

Additional example script packages

In addition to the script packages provided with the system, you can review the following examples that demonstrate how script packages are used in a virtual system environment.

Example script package to install an application

This script installs an existing application in a user-specified location. The location can be either on the file system, or at some remote location. The wsadmin tool installs the application. You can pass in installation arguments during deployment.

Script variables

There are two parameters that are part of this script package.

APP_LOCATION:
Required. The location of the application. The location of the application can be either a file system location or remote location over http or https.

INSTALL_ARGS:
Optional. Install arguments for the AdminApp.install() wsadmin command. The default command is "AdminApp.install(appLocation, ['-usedefaultbindings'])". Other arguments can be supplied using this variable. An example value for this variable is "-usedefaultbinding
-server myServer -appName MyApp. If the application is remote, it is copied to the current working directory before the installation command is started.

cb.script.json example

```json
[
  {
    "name": "Install application",
    "version": "1.0.0",
    "description": "This script package installs the specified application",
    "command": "/bin/sh ${WAS_PROFILE_ROOT}/bin/wsadmin.sh",
    "log": "${WAS_PROFILE_ROOT}/logs/wsadmin.traceout",
    "location": "/opt/tmp/installapp",
    "timeout": "0",
    "ostype": "linux/unix",
    "commandargs": "-lang jython -f /opt/tmp/installapp/install_app.jy $APP_LOCATION $INSTALL_ARGS",
    "keys": [
      {
        "scriptkey": "APP_LOCATION",
        "scriptvalue": 
      },
      {
        "scriptkey": "INSTALL_ARGS",
        "scriptvalue": 
      }
    ]
  }
]
```

Example script

**Note:** This example script is designed for version 7.0.0.x patterns only.

```python
import urllib
from java.io import File
from java.lang import Boolean
from java.lang import String
from java.net import URL

def download(url):
    fileLocs = String(url).split('/')
    lastPart = fileLocs[len(fileLocs) - 1]
    file = File(lastPart)
    file.createNewFile()
    newFileLoc = file.getAbsolutePath()
    urllib.urlretrieve(url, newFileLoc)
    return newFileLoc

def copyZip(binURL):
    binURL = str(binURL)
    url = None;
    fileRemote = Boolean.FALSE
    appFileLoc = ''
    try:
        url = URL(binURL)
        fileRemote = Boolean.TRUE
    except:
        pass
    if fileRemote:
        print 'Start retrieval of ' + binURL

Chapter 7. Managing and deploying virtual patterns  381
Example script to configure the trace levels

This script package sets a trace specification level (example "com.ibm.ws.ibm.*=info") on all servers in a cell. It can be included on either a stand-alone pattern part or a Deployment Manager pattern part. Users can specify the trace specification during deployment.

Script variables

The following parameter is included in this script package.

**TRACE_SPEC:**

Specifies the trace specification for the cell. This parameter is required.

**cbscript.json example**

```
[
  {
    "name": "Configure Trace Specification",
    "version": "1.0.0",
    "description": "This script package configures trace specification on all servers in a cell",
    "command": "${WAS_PROFILE_ROOT}/bin/wsadmin.sh",
    "log": "${WAS_PROFILE_ROOT}/logs/wsadmin.traceout",
    "location": "/opt/tmp/configtrace",
    "timeout": "0",
    "ostype": "linux/unix",
    "commandargs": "-lang jython -f /opt/tmp/configtrace/configure_trace.jy $TRACE_SPEC",
    "keys":
      [
        {
          "scriptkey": "TRACE_SPEC",
          "scriptvalue": "",
          "scriptdefaultvalue": ""
        }
      ]
  }
]
```

**Example script**

**Note:** This example script is designed for version 7.0.0.x patterns only.

```
from java.lang import String
traceSpec = sys.argv[0]
nodes = AdminNodeManagement.listNodes()
for node in nodes:
    nodeStr = String(node)
```
Example script package to create a server

This script creates an application server on all of the custom nodes in an IBM
Cloud Orchestrator pattern or on the node part for which it is included. The script
package is intended to be used on either a Deployment Manager or stand-alone
server part in a pattern. The name of the application server is specified by the user.

Script variables

The following parameter is include in this script package.

SERVER_NAME:
  Specifies the name of the server to be created on each node. If multiple
  nodes exist in the pattern, the server name is augmented with a counter
  that begins at 1. This parameter is required.

Example script

Note: This example script is designed for version 7.0.0.x patterns only.

serverName = sys.argv[0]
managedNodeStr = AdminTask.listManagedNodes()

if len(managedNodeStr) != 0:
    managedNodes = managedNodeStr.split("\n")
i=1
for managedNode in managedNodes:
    thisServer = serverName + " " + str(i)
    AdminServerManagement.createApplicationServer(managedNode, thisServer, 'default')
Example script package to create a WebSphere Application Server cluster

This script package can be included on a Deployment Manager part. It creates an application server on each custom node in a cell and creates an application server cluster from those servers. Both the cluster and server names can be specified during deployment.

Script variables

There are two parameters included in this script package.

**CLUSTER_NAME:**

Specifies the name of the new cluster. This parameter is required.

**SERVER_NAME:**

Specifies the name of the servers. The script package automatically appends numbers to the supplied server name to ensure that each server name is unique. This parameter is required.

cbscript.json example

```json
[
  {
    "name": "Cluster creation",
    "version": "1.0.0",
    "description": "This script package creates a server on each node within the cell and then creates a cluster from those servers",
    "command": "${WAS_PROFILE_ROOT}/bin/wsadmin.sh",
    "log": "${WAS_PROFILE_ROOT}/logs/wsadmin.traceout",
    "location": "/opt/tmp/createcluster",
    "timeout": "0",
    "ostype": "linux/unix",
    "commandargs": "-lang jython -f /opt/tmp/createCluster/createCluster.jy $CLUSTER_NAME $SERVER_NAME",
    "keys":
      [
        {
          "scriptkey": "CLUSTER_NAME",
          "scriptvalue": "",
          "scriptdefaultvalue": ""
        },
        {
          "scriptkey": "SERVER_NAME",
          "scriptvalue": "",
          "scriptdefaultvalue": ""
        }
      ]
  }
]
```

Example script

*Note:* This example script is designed for version 7.0.0.x patterns only.
cellName = AdminControl.getCell()
cellName = sys.argv[0]
serverName = sys.argv[1]
managedNodeStr = AdminTask.listManagedNodes() 
managedNodes = managedNodeStr.split("n")
i=0
for managedNode in managedNodes:
    appServers = AdminServerManagement.listServers('APPLICATION SERVER', managedNode)
    webServers = AdminServerManagement.listServers('WEB_SERVER', managedNode)
    appSrvLen = len(appServers)
    webSrvLen = len(webServers)
    if appSrvLen == 0 and webSrvLen == 0:
        AdminTask.createCluster(['-clusterConfig [-clusterName ' + clusterName + ' -preferLocal true]'])
        cluster = AdminConfig.getId('/ServerCluster:' + clusterName + '/')
        memberName = serverName + str(i)
        node = AdminConfig.getId('/Node:' + managedNode + '/')
        AdminConfig.createClusterMember(cluster, node, [[memberName, memberName]])
        i = i + 1
    AdminConfig.save()

Managing add-ons

You can configure user and NIC parts in your catalog and then use them as parts in your patterns.

Before you begin

A basic set of add-ons are provided by IBM and must be downloaded before these steps are performed. For more information, see "Add-ons in the catalog."

Add-ons in the catalog

Use the console to manage virtual system pattern add-ons in the catalog. You can use the default add-ons provided with the system, or you can clone and modify them, or create and configure your own. You can then add them as parts to one or more virtual system patterns.

Add-ons are specialized scripts that customize your virtual machine configuration. Add-ons provide fine tuning for hardware and operating system configurations. A set of basic add-ons with an initial configuration is provided. You can use these default add-ons, clone and modify them as needed, or create new ones.

Add-ons are implemented at the following levels:
- Deployment logic provisions new hardware for disks and network interface controllers (NICs) that cannot be customized.
- Deployment logic configures the provisioned resources. Resources are packaged as default implementation scripts that are fully customizable. For example, the Default add user add-on has a defaultadduser.zip package that contains a script to define a user account.

To view the list of add-ons that are available in the catalog, click PATTERNS > Pattern Design > Add-Ons.
After creating an add-on, it can be dropped onto a topology pattern part from the Virtual System Patterns window. When the pattern is deployed, you provide the customization parameters for the add-ons to customize the hardware and operating system configuration.

The following set of default add-ons are provided with the product:

**Default add NIC**
Adds a new virtual network interface controller (NIC) to the virtual machine, configures its IP address information, and activates it. Use this add-on for virtual image parts that support communication using ssh to run scripts after initial activation of the virtual machine.

*Note:* This add-on is not supported on PowerVM virtual images.

**Default configure NIC**
Triggers configuration via VSAE of the additional NICs present in the image.

*Note:* This add-on is not supported on PowerVM virtual images.

**Default add disk**
Adds a virtual disk to the virtual machine and optionally formats and mounts the disk. Prerequisite: the `parted` (parted RPM) and `sfdisk` (util-linux RPM) tools must be installed for a RedHat image (or other type of packages depending on different operating systems). The prerequisite for VMware virtual images is that the virtual disk type must be SCSI.

**Default AIX add disk**
Adds a virtual disk to the virtual machine and optionally formats and mounts the disk. Use this add-on for PowerVM virtual images.

*Note:* IBM Cloud Orchestrator does not support the disk add-on function for PowerVM with Shared Storage pool.

**Default Windows add disk**
Adds a virtual disk to the virtual machine and formats and makes the disk available under new letter. Prerequisites: PowerShell and Diskpart tools, which are available in the default Windows installation. New disk is partitioned using MBR schema. The prerequisite for VMware virtual images is that the virtual disk type must be SCSI.

**Default raw disk**
Adds a virtual disk to the virtual machine, the disk is added raw without partitions or formatting. For PowerVM virtual images, you must run the `cfgmgr` to refresh the system configuration and to see the new disk. The prerequisite for VMware virtual images is that the virtual disk type must be SCSI.

*Note:* IBM Cloud Orchestrator does not support the disk add-on function for PowerVM with Shared Storage pool.

**Default add user**
Defines an additional user on the virtual machine. The default add-on script runs a simple add user command. No additional account configuration is performed.
Managing add-ons with the user interface

You can manage add-ons in the catalog with the user interface and then add them to deployable patterns.

Before you begin

You must be assigned the **catalogeditor** role or the **admin** role to perform these steps.

About this task

You can use, add, modify or delete the following types of add-ons in your catalog:
- Disk
- NIC
- User

Procedure

1. Navigate to the Add-Ons window by clicking **PATTERNS > Pattern Design > Add-Ons** from the menu bar.
2. You can perform the following tasks:
   - “Adding add-ons to the catalog” on page 388
   - “Cloning an add-on” on page 389
   - “Editing an add-on” on page 390
   - “Making add-ons read-only” on page 390
   - “Associating an add-on with a pattern” on page 391
   - “Deleting an add-on” on page 392

For more information about the fields on the Add-Ons window, see “Fields on the Add-Ons user interface” on page 393.

Results

After completing these steps you have managed the add-ons that you can add to parts on deployable patterns.

What to do next

After you have configured your add-ons you can work with them as parts on a pattern in the Pattern Editor window. For more information about editing parts, see “Configuring parts” on page 423. For more information about working with patterns, see “Working with virtual system patterns (classic) in the user interface” on page 412. If you are working with a NIC add-on, it can be configured with an environment profile. See “Managing environment profiles” on page 344 for more information.
Adding add-ons to the catalog
You can add new add-ons in addition to the add-ons already included in the product.

Before you begin
You must be assigned the catalogeditor role or the admin role to perform these steps. A basic set of add-ons are provided by IBM and must be downloaded before these steps are performed.

About this task
You can either clone an existing add-on or you can create a new add-on. To clone an existing add-on, either a default add-on or one that has been added by a user, see “Cloning an add-on” on page 389. To add a new add-on to the catalog, use the following steps.

Procedure
1. From the left panel of the Add-Ons window, click Create New to add an Add-On to the catalog.
2. Select the add-on archive and click Import. The new add-on displays in the left panel of the Add-Ons window. The right panel displays configuration options for the add-on.
3. Select the type of add-on. Add-ons can be one of the following types:
   - Disk Adds a virtual disk to the virtual machine and, optionally, formats and mounts the disk.
   - NIC Adds a virtual network interface controller (NIC) to the virtual machine. NIC add-ons are deployed with environment profiles.
   - User Defines an additional user on the virtual machine.
4. Optional: Add a description to help identify the add-on.
5. Optional: In the Add-on package files section, provide the customized add-on package file with the Browse field and the Upload button.
6. Optional: Use the Environment section to add environment variables.
7. Optional: Specify the standard script package parameters for the following directories:
   - Working
   - Logging
   - Executable
   - Arguments
8. Optional: Set a timeout value in the Timeout field.
9. Optional: Add or change the user permissions for this add-on with the Access granted to field.

Results
You added a new add-on to the catalog.

What to do next
The add-on is ready to be used as a part in your patterns.
Cloning an add-on

You can create an add-on based on an existing add-on. Default add-ons are included that can either be added as they are or cloned and edited. The cloned add-on can then be modified to your specifications.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

About this task

When you clone an add-on, the copy is pre-populated with the configuration data from the original add-on. Use this task to copy an existing add-on that can then be edited.

Procedure

1. Locate the add-on to clone. If it is not displayed, you can search for the add-on. From the left panel of the Add-Ons window, enter all or part of the name of the add-on in the search field.
2. Select the add-on to clone. Click the add-on to copy from the listing on the left panel of the Add-Ons window. Details about the add-on are displayed in the right panel of the Add-Ons window.
3. Click the clone icon to copy the add-on you have selected.
4. Enter the name for the new add-on and click OK. The details panel for the new add-on is now displayed and can be modified. When you have completed the configuration for the add-on, the add-on is saved for you as you navigate away from the Add-Ons window.
5. Optional: Add or change the description to help identify the add-on.
6. Optional: In the Add-on package files section, provide the add-on package files in one of the following ways:
   • Provide a custom add-on package using the browse function to locate your custom package.
   • Download and modify the default add-on implementation.
7. Optional: Use the Environment section to remove existing environment variables or create new ones.
8. Optional: Configure standard script package parameters for the following directories:
   • Working
   • Logging
   • Executable
   • Arguments
9. Optional: Set a time out value in the Timeout field.
10. Optional: Add or change the user permissions for this add-on.

Results

After you have completed these steps, any users or groups with access, can use the add-on with pattern nodes.
What to do next

You can now associate this add-on with a pattern. See “Working with virtual system patterns (classic)” on page 411 for more information about patterns.

Editing an add-on

You can edit any add-on that is not read-only. You can modify a add-on to suit the changing needs of your environment.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

About this task

You can edit any add-ons that are not read only and to which you have write access. This task provides information about editing existing add-ons.

Procedure

1. Select the add-on. Select the add-on you want to edit from the left panel of the Add-Ons window. Add-ons that have the locked symbol by them cannot be edited. Add-ons with the edit symbol beside them can be edited. The information about that add-on is shown in the right panel of the Add-Ons window.
2. Click the edit icon. From the top of the right panel of the Add-Ons window, click the edit icon.
3. Edit the fields on the right panel of the Add-Ons window. For more information about these fields, see “Fields on the Add-Ons user interface” on page 393.

Results

When you have finished editing an add-on, it is ready to be added to part on a pattern.

What to do next

You can lock the add-on against future editing. For more information, see “Making add-ons read-only.” You can add the add-on to a part on a pattern. For more information, see “Working with virtual system patterns (classic)” on page 411.

Making add-ons read-only

Either draft or read-only add-ons can be deployed for testing or production, but making an add-on read-only prevents further edits. Making add-ons read-only provides consistent reuse in the cloud.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

About this task

You can make an add-on read-only to prevent further editing to the add-on.
Procedure
1. Select the add-on. From the left panel of the Add-Ons window, select the add-on. Add-ons that have the read-only symbol by them are already read-only and cannot be edited. Add-ons with the edit symbol beside them are not read-only and can be edited. Basic information about the selected add-on is shown in the right panel of the Add-Ons window.
2. Made the add-on read-only to lock it to future editing. Click the Lock icon in the upper right toolbar of the Add-Ons window.
3. Verify that you want to make the add-on read-only. When prompted to verify that you want to make the add-on read only, click OK to lock the add-on.

Results
When you have made the add-on read-only, it can be cloned or deleted but it cannot be edited.

What to do next
You can include the add-on on a part in a pattern to be deployed to the cloud. For more information, see “Working with virtual system patterns (classic)” on page 411.

Associating an add-on with a pattern
You can associate an add-on with a pattern and defined cell through the user interface.

Before you begin
The add-on must be configured. For more information about configuring the add-on, see “Editing an add-on” on page 390.

About this task
Use this task to associate an add-on with a pattern.

Procedure
1. Navigate to the Virtual System Patterns window by clicking PATTERNS > Instances > Virtual System Patterns, for virtual systems, or by clicking PATTERNS > Instances > Virtual System Patterns (Classic), for virtual systems (classic).
2. Select a pattern. In the left panel of the Virtual System Patterns window, select the pattern with which to associate the add-on. The pattern must not be read-only. Basic information about the selected pattern displays in the right panel of the Virtual System Patterns view.
3. Edit the pattern. Click Edit on the upper right of the Virtual System Patterns (Classic) window, for virtual systems (classic), or click Open on the upper right of the Virtual System Patterns window, for virtual systems.
4. Add Add-Ons.
   For virtual systems (classic): from the drop-down box in the left panel of the Pattern Editor, click Add-Ons. A list of the add-on parts is provided that can be dropped into the virtual image parts on the right panel of the Pattern Editor view. This list can contain any add-ons provided for use with IBM Cloud Orchestrator. Add-ons can then be added to the virtual image parts. You can
drop any add-on from this list onto the virtual image parts on the canvas on the right. This associates the add-on with that part.
For virtual systems: in Pattern Builder click the image object. Then click **Add a Component Add-On** and select from the list the add-on that you want to add to the image object. This operation associates the add-on with the image part.

5. Optional: Configure any properties defined in the add-on. Properties added to add-ons can be defined when associating the add-on with a part or it can be defined during deployment.
For virtual systems (classic): click the edit properties icon to set the value now. It is possible to use a variable syntax to set the value for properties where the value is not yet known. For more information about setting the value of a property to be variable, see “Properties variable syntax” on page 367.
For virtual systems: click the associated add-on object in the image part and edit the properties of the add-on on the menu in the right side of Pattern Builder.

**Results**

You added one or more add-ons to the virtual images on the pattern.

**What to do next**

When you have associated the add-on with a pattern, you can complete configuration of the pattern and deploy it to the cloud group.

**Deleting an add-on**

You can manually delete an add-on, disassociating it with the pattern and cell, using the user interface.

**Before you begin**

You must be assigned the **catalogeditor** role and be granted **all** access to the add-on you want to remove. You can also perform these steps if you are assigned the **admin** role. Add-ons that are read-only, or that are in use, cannot be deleted. Before you delete an add-on, ensure that it is no longer needed by the pattern and the cell with which it is associated.

**About this task**

This task provides information about manually deleting an add-on from the Add-On window of the IBM Cloud Orchestrator user interface.

**Procedure**

1. Locate the add-on. If it is not displayed, you can search for the add-on you want to delete. From the left panel of the Add-Ons window, enter all or part of the name of the add-on in the search field.
2. Select the add-on to delete from the listing on the left panel of the Add-Ons window. Details about the add-on are displayed in the right panel of the Add-Ons window.
3. Determine if it can be deleted. An add-on can only be deleted if it is not:
   • Marked as read-only. The read-only icon is shown in the listing of add-ons if it is read-only.
• Included in any patterns. If it is included in any patterns, the delete icon is not available and the **Included in patterns** field displays the linked patterns in which this add-on is included.

If the add-on is referenced by any patterns, you can click the pattern name link in the **Included in patterns** field to go to the Virtual System Patterns panel for that pattern. From this panel, you can remove the add-on part from the part, or parts, on the pattern.

4. Delete the add-on. Click the delete icon on the top right of the Add-Ons window.

5. Confirm the deletion. You are prompted to confirm that you want to delete the selected add-on. Click **OK** to delete the add-on.

**Results**

The add-on is deleted from IBM Cloud Orchestrator.

**Fields on the Add-Ons user interface**

You can manage the add-ons in your catalog using the fields on the Add-Ons window of the IBM Cloud Orchestrator user interface.

To work with add-ons with the IBM Cloud Orchestrator user interface you can use the Add-Ons window.

**The left panel**

The left panel of the Add-Ons window provides the following options to work with add-ons:

**Create New**

Click **Create New** to open a window in which you can define your new add-on.

**Search**

Enter the name of an add-on in this field to search for it. You can use the up and down arrow keys to sort through the listing of add-ons.

**List of add-ons**

The listing of add-ons contains the default add-ons that are already defined to IBM Cloud Orchestrator, by name.

To work with an add-on, first select it from the list in the left panel of the Add-Ons window.

**Icons on the upper right**

The upper right of the Add-Ons window provides the following icons:

**Refresh**

Refreshes the status of the add-ons and updates the fields on the Add-Ons window.

**Clone**

Creates a copy of the add-on, even a locked add-on, that can be edited.

**Lock**

Makes the add-on read-only and locks it against further editing.

**Delete**

Removes the add-on from the IBM Cloud Orchestrator catalog.
The right panel

Selecting an add-on shows the name of the add-on at the top of the right panel and the following fields on the right panel:

Description
The description of the add-on. You can edit this description to provide meaningful information about the add-on.

Type
The add-on can be one of the following types:
- Disk
- NIC
- User

The type of add-on is selected when the add-on is created and this field cannot be changed after creation of the add-on.

Created on
The creation time of the add-on, as the number of seconds since midnight January 1, 1970 UTC. When the add-on is displayed, this value is shown as the date and time in the local timezone.

Current status
The status of the add-on can be one of the following status types:

The edit icon
The add-on can be edited.

Read-only
The add-on is locked to editing. For information about making an add-on read-only, see “Making add-ons read-only” on page 390.

Updated on
The time the add-on was last updated, as number of seconds since midnight, January 1, 1970 UTC. When the add-on is displayed, this value is shown as the date and time in the local timezone. This field is read-only.

Add-on package files
If you are cloning one of the provided default add-ons, you can create custom add-ons by downloading and modifying the add-on package. The add-on package is defined for each type of add-on:

Default add NIC
Download the defaultaddnic.zip package.

Default configure NIC
Download the defaultconfigurenic.zip package.

Default add disk
Download the defaultadddisk.zip package.

Default AIX add disk
Download the defaultaixadddisk.zip package.

Default Windows add disk
Download the defaultwinadddisk.zip package.

Default raw disk
Download the defaultaddrawdisk.zip package.

Default add user
Download the defaultadduser.zip package.
The Download link beside existing script package zip files enables these packages in the catalog to be loaded.

For new add-ons, you can upload a custom script package by clicking the Browse field. The Browse field opens a system file upload window to browse to the location of the package to load. After locating the package, clicking the Upload button loads the add-on package to your catalog.

Environment
The environment property holds the add-on keys and default values for the add-on. What this field contains depends on the type of add-on and the situation in which you are viewing it:

- If you are creating a new add-on, you can define the environment variables. The Add variable field, with the name and value entry fields, is used to list the environment variables to add. The Add link adds the provided environment variable to the add-on.
- If you are viewing an add-on that is read-only which has environment variables set, those environment variables are shown.
- If you are working with a NIC add-on, there are no environment variables that can be set. A NIC add-on must be deployed using an environment profile.

Working directory
The directory, on the virtual machine, into which files for this add-on package are to be placed.

Logging directory
The directory, on the virtual machine, that is to contain the log files generated by this add-on.

Executable
Specifies the command to be invoked for this add-on package. The executable can be a command already on the virtual machine (for example wsadmin, tar, ant, or another system command). You can also provide your own script to be run as part of the script package.

Arguments
The field to enter any arguments that are to be passed to the command, the command line that is passed to the executable command. This field can optionally contain environment variables and other valid command-line syntax. You can also access environment using standard techniques (for example shell, or ant.)

Timeout
The maximum amount of time to wait for this add-on to finish running on a virtual machine. Specify the timeout as the number of milliseconds to wait, or 0 to wait indefinitely for the add-on to complete.

Access granted to
The access control list for this add-on. Users or projects who have access this add-on are listed in this field as links. The Add more... entry field enables the owner or anyone with proper access to the add-on to provide access for more projects.
Working with virtual system patterns

Use the Self-service user interface to manage virtual system patterns in IBM Cloud Orchestrator.

You can manage virtual system patterns in the Virtual System Patterns and Pattern Builder pages in the Self-service user interface. Virtual system patterns implement deployment topologies that are created using one or more virtual images and applications in the system catalog.

You can use the virtual system patterns provided by IBM Cloud Orchestrator without modification and deploy them to the cloud. You can also create new patterns, or make a copy (clone) of an existing pattern, and use the Pattern Builder to modify the pattern to alter the topology and configuration of your environments.

Creating virtual system patterns

Use the Self-service user interface to create virtual system patterns.

Before you begin

You must be granted access to patterns, or access to create patterns, to complete this task.

About this task

Use the Pattern Builder to create virtual system patterns. When you start the Pattern Builder, it opens in a separate window. If you are using Microsoft Internet Explorer Version 7 or Version 8, you might need to change your browser settings so that the Pattern Builder is displayed correctly. Specifically, configure pop-up windows to open in a new tab in the current window. From the browser menu, click Tools > Internet Options.

You can use the Self-service user interface, the command-line interface, or the REST API to complete this task. For the command-line and REST API information, see the related links.

Procedure

1. Click PATTERNS > Pattern Design > Virtual System Patterns.
2. Click Create New.
3. Build your virtual system pattern:
   a. Select a virtual system template or start from a blank pattern.
   b. Set the Name and Version for the virtual system pattern.

   Note: Use a unique name and version combination for the pattern. If you attempt to create a pattern with the same name and version as an existing pattern, an error is displayed.

   c. Click Start Building.

   The Pattern Builder opens in a new web browser page where you can build the virtual system pattern.

4. Specify the properties for the pattern in the pattern properties pane:

   Name  The name of the virtual system pattern.
Version
The version of the virtual system pattern.

Description
The description of the virtual system pattern. This field is optional.

Type
Leave Pattern selected to create a virtual system pattern or select Pattern Template to create a virtual system template that is used as the basis for creating other virtual system patterns.

Lock option for plugin usage
Specify how this virtual system pattern is affected by upgrades to the pattern type or to IBM Foundation Pattern.

Unlock plugins
If the pattern type is upgraded, use the latest versions of pattern type plug-ins. If IBM Foundation Pattern is upgraded, use the latest version.

Lock all plugins
Do not change the version of plug-ins or the version of the IBM Foundation Pattern that is associated with this virtual system pattern when an upgrade occurs.

Lock all plugins except Foundation plugins
If the pattern type is upgraded, do not change the version of the plug-ins that are associated with this virtual system pattern. If IBM Foundation Pattern is upgraded, use the latest version.

Note: If you select Lock all plugins or Lock all plugins except Foundation plugins, you can view a list of which plug-ins are locked. Click the Source tab in Pattern Builder. The application model source is displayed. Search for the element plugins to view the list.

5. If you selected a blank template, the canvas is empty and you can start building the virtual system pattern. If you selected a template, customize the virtual system pattern:
   a. Drag the assets that you want to add to the virtual system pattern onto the canvas.

Note:
• When you add an asset to the canvas, the properties for that asset are displayed in the right pane. The properties vary depending on the asset that you add. Required properties are denoted by a red asterisk. If a required property is not set, a warning icon displays on the asset in the canvas.
• The right pane displays the properties for the asset that is selected on the canvas. You can access the settings for an asset by selecting it on the canvas.
• Some properties have a lock symbol next to their value in the right pane. If you lock a property by clicking this symbol, that property is not configurable from the deployment page when a user deploys the virtual system pattern.
• Some properties have an icon with a green arrow next to them. Click this icon to add a reference to a component-level or pattern-level
parameter as the value for the attribute, and create an explicit data dependency link. After you click the icon, the Add a reference page is displayed:

- Select whether you want to reference a component-level parameter or a pattern-level parameter.
- Then, select the component and an output attribute for that component that you want to reference.
- Click **Add** to create the reference.

**Tip:**

- For example, if your pattern contains a DB2 component and a WebSphere Application Server component, you might want the `db_user` parameter for the DB2 component to reference the `WASROOT` output attribute of the WebSphere Application Server component.
- Refer to the documentation for the pattern that you are using for specific details about its components and attributes.

- If you want to remove an asset from the canvas, click **Remove component** on the asset.
- After you build your pattern, you can change to the **List View** tab to view the topology as a vertical list of components. You can configure the values for the properties in the pattern from the canvas, or from the **List View** tab.

**Images**

Virtual images provide the operating system and product binary files that are required to create a virtual system instance. Some virtual images include only the operating system files, while others also include product binary files.

If multiple versions of the image are available, such as 8.5.5.0 and 8.5.5.1, select the version that you want to deploy with the pattern after you add the image to the canvas.

**Scripts**

Scripts can include almost any set of executable files and artifacts that are valid for the target virtual machine environment. Create these scripts on your local workstation, in IBM PureApplication® System, or by using the Plug-in Development Kit. Then, import the script package to the system so that it is available in the catalog and in the **Assets** palette in the Pattern Builder.

**Note:**

- Scripts cannot be added to the canvas directly. You must add them to images or components.
- If multiple versions of the script are available, such as 1.0 and 2.0, select the version that you want to use with the pattern after you add the script to the canvas.
- You cannot add scripts to some images, such as the DataPower® image. If an image does not support scripts, a message displays on the canvas when you attempt to add the script to the image.

For more information about creating script packages, see the Related information section.

**Software Components**

Software components install, and possibly configure software on
the virtual system pattern. You can drag a software component onto a virtual image that is on the canvas. If you drag a software component directly onto the canvas, it is automatically added to a virtual image that meets the requirements for that software component.

If multiple versions of the software component are available, select the version that you want to deploy with the pattern after you add the software component to the canvas.

b. Edit the connections between assets, as needed. To add a link between two assets, hover over one of the assets until the blue circle turns orange. Helper text describes the asset that is associated with the link. Select the circle, drag a connection to the second asset until the asset is highlighted, and then release. When you create a link between two assets, you can map a property of one asset to a property of the asset to which it is linked.

For example, you might have one image with a software component that requires a DB2 server, and another image with a software component that installs a DB2 server. You can link the software component to the DB2 server. When you create the link, you are prompted to map the properties between the two assets. You can map the **DB2 server IP address** property for the first software component to the **Server IP address** property for the second software component.

c. Add policies to the pattern or a component, as needed.

You can apply a policy globally at the pattern level, or apply it to a specific component that supports the policy. When you apply a policy globally, it is applied to all components in the virtual system pattern that support it. If you apply a policy to a specific component and also apply it to the whole virtual system pattern, the configuration of the component-specific policy overrides the pattern level policy.

To add policies to the virtual system pattern, click **Add policy to pattern** and select a policy, or select a component part on the canvas and click the **Add a Component Policy** icon to add a component-specific policy. You can add these policies to the pattern or a component:

**iFix policy**

Add an interim fixes policy to install emergency fixes during deployment to the pattern or to a particular virtual machine, depending on whether you apply the policy at the pattern-level or component-level. After you add the interim fix policy, choose one or more fixes from the list to apply at deployment time. The fixes that display in the list are emergency fixes that were uploaded to the emergency fixes catalog (through **Catalog > Emergency Fixes**) or to the Installation Manager repository (through **Catalog > IBM Installation Manager Repository**).

Interim fixes in the Emergency Fixes catalog can be configured to be applicable to an image, plug-in, or middleware. Only fixes that are applicable to an image, plug-in, or middleware that is in the pattern are displayed.

Interim fixes in the IBM Installation Manager repository can be configured to be applicable to middleware. Only fixes that are applicable to middleware that is in the pattern are displayed.

**Security policy**

Add a security policy to configure whether Secure Shell (SSH) password login is available. If you configure this policy at the
component-level, SSH log-in is disabled for the virtual machine that deploys for the image where the policy is set. If you configure this policy at the pattern-level, SSH log-in is configured for all virtual machines that deploy as part of the pattern.

**Base Scaling Policy**

Scaling is a runtime capability to automatically scale your virtual system instance as the load changes. A scaling policy component defines this capability and the conditions for processor and memory, under which scaling activities are performed.

Specify the following attributes for the scaling policy:

**Number of instances**

Specifies the number of cluster members that are hosting the instance. The default value is 1. The acceptable value range is 2 - 10. This attribute is required.

**Instance number range of scaling in and out**

Specifies the scaling range for instance members that are hosting the topology. The default range for this attribute is 1 - 10. The acceptable value range is 1 - 50. This attribute is required.

**Maximum vCPU count per virtual machine**

Specifies the maximum number of virtual processors that are used by the topology.

The acceptable value range is 2 - 16.

**Maximum memory size per virtual machine**

Specifies the maximum amount of virtual memory, in gigabytes, that are used by your application.

**CPU Based**

**Scaling in and out when CPU usage is out of threshold range**

Specifies the processor threshold condition to start scaling activity. When the average processor use of a virtual machine is out of this threshold range, the specified scaling action is taken. If the policy is set at the pattern level, scaling actions are taken if the processor use of any virtual machine in the topology is out of the threshold range. If the policy is set at the component level, scaling actions are taken if the processor use of the virtual machine where the component is running is out of the threshold range. The acceptable value range is 0 - 100%.

**Minimum time (seconds) to trigger add or remove**

Specifies the time duration condition to start scaling activity. The default value is 300 seconds. The acceptable value range is 30 - 1800. This attribute is required.

**scaling by**

Specifies the action to take when scaling is triggered. You can select add vCPU only, add or remove nodes, or add or remove nodes; add vCPU first and then add nodes.
Memory Based

Scaling in and out when memory usage is out of threshold range

Specifies the memory threshold condition to start scaling activity. When the average memory usage of your topology is out of this threshold range, the specified scaling action is taken. The default value is 20 - 80%. The acceptable value range is 0 - 100%.

Minimum time (seconds) to trigger add or remove

Specifies the time duration condition to start scaling activity. The default value is 300 seconds. The acceptable value range is 30 - 1800. This attribute is required.

then scaling by

Specifies the action to take when scaling is triggered. You can select add virtual memory only, add or remove nodes, or add or remove nodes; add virtual memory first and then add nodes.

Note: If the middleware requires a restart before its virtual memory can be increased, such as with IBM WebSphere Application Server, the middleware is restarted in a rolling method so that there is no downtime during the scaling.

Note: You can select CPU-based scaling, memory-based scaling, or both. If you select more than one type, the relationship between the types is handled by the system as an OR. For example, if you select processor and memory-based scaling, with the action to add or remove a node, this action is triggered if the condition set for either processor or memory scaling is met.

Routing policy

Add a routing policy to configure the topology or component for Elastic Load Balancing (ELB). After you add the policy, specify the following attributes for the routing policy:

Endpoint

The endpoint is the URL that is used to access the application through Elastic Load Balancing.

Port

Enter the port that is used by the topology (for a pattern-level policy) or component (for a component-level policy). This port is opened on the firewall when the pattern deploys.

Enable HTTPS

Specify whether HTTPS is used by the topology (for a pattern-level policy) or component (for a component-level policy). If you select Enable HTTPS, you must enter the port that the topology or component uses for HTTPS. If the port is specified, it is opened on the firewall when the pattern deploys.
6. Optional: Click the **Add a Component Add-on** icon on the image to add an add-on. For more information about add-ons, see the **Related concepts** section.

7. Optional: Click **Advanced Options** on the canvas to configure advanced options. The advanced options that are available depend on the topology of the virtual system pattern you are editing.

**Note:** When you open the advanced options editor for a new virtual system pattern that has no advanced options set, the displayed settings are the recommended values for the topology.

8. Optional: If your pattern includes a WebSphere Application Server software component, configure these properties:

   - Specify the location of the WebSphere Application Server installation.

   **Note:** This property is mapped to the installation directory property of the WebSphere Application Server software component, so it should be populated automatically. If it is not populated, enter the installation location manually.

   - If there is more than one WebSphere Application Server on the node, and you want to specify the order in which the servers are restarted, specify the order in the **WAS Servers Restarting Order** property. Enter the server names in the required order. Separate the server names with semicolons.

   - Specify the WebSphere Application Server user name.

9. Set the order of deployment for each of the components in the pattern.

   a. Use the **Move up** and **Move down** options on software components and scripts to determine the order in which components are started.

   b. Change to the **Ordering** tab to change the order for other components in the pattern. You can also modify the order for software components and scripts from this tab.

10. Click **Save**.

**What to do next**

When creating virtual system patterns, you can use the Pattern Builder to specify an OS prerequisite list or a set of global package installation options that are required by a component in the metadata of the system plug-in. The Red Hat Update Infrastructure (RHUI) Service or the Red Hat Satellite Service must be deployed in the cloud group before you deploy the pattern, which allows virtual machine clients to download and install the RPMs using these services. The RHUI Service provides a Yellowdog Updater, Modified (YUM) repository.

If you are a software component developer and need to ensure certain Red Hat OS RPMs are installed in the image, you must ensure that they are described in the metadata.json file. For example, to add MySQL and VNC packages to the deployed virtual machines automatically, you can add a stanza to the metadata.json file as shown in the following text:

```json
"configurations": {
   "partsKey": "Yum",
   "packages": [
      "YUM"
   ],
   "prereq_os_packages": [
      ""}
```
During pattern deployment, OS packages can be added or removed from the list specified by the Pattern Builder, and can override deployment policies at a virtual machine level or a pattern level for the entire pattern with multiple virtual machines.

**Importing virtual system patterns**

You can import virtual system patterns so that you can edit and deploy the imported topology for use in your system.

**Before you begin**

You must have access to patterns, or access to create patterns, to complete this task.

**About this task**

You can import virtual system patterns that were previously exported. The version for the pattern is retained when you import the pattern. You can use the Self-service user interface or the command-line interface to complete this task. If you use the Self-service user interface to import a pattern, you can import only the pattern model. To import pattern artifacts (script packages, add-ons, images, or software packages) along with the pattern, use the command-line interface instead.

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual System Patterns**.
2. Click the **Import** icon on the toolbar.
3. Click **Browse** to select the compressed file that contains the pattern that you want to import.
4. Click **Import**.
   - If there are no existing patterns on the system with the same name and version as the pattern that you are importing, the pattern is imported.
   - If a pattern with the same name and version exists on the system, and one or both of the patterns are not read-only, you are prompted with options:
     a. Specify a unique name for the imported pattern.

     **Note:** This option is available only when the pattern that you are importing is not read-only.

     b. Specify a unique version for the imported pattern.

     **Note:** This option is available only when the pattern that you are importing is not read-only.

     c. Replace the existing pattern.

     **Note:** This option is available only when the existing pattern is not read-only.

After you make a selection, click **Import** again to complete the import process.
Cloning virtual system patterns

You can clone an existing virtual system pattern and then customize the copy for your environment.

Procedure

1. Click PATTERNS > Pattern Design > Virtual System Patterns.
2. Select a virtual system pattern.
3. Click the Clone icon on the toolbar.
4. Provide the following information about the new virtual system pattern:
   - **Name**: Enter a unique name for the virtual system pattern in the Name field.
   - **Virtual images**: Select a different virtual image than the image that is used in pattern for any of the virtual images that are in the pattern as needed.
5. Click OK. When the information is processed, the Virtual System Patterns page displays. The virtual system pattern that you created is added to the list. Select the cloned virtual system pattern to display more details about the pattern.
6. Click the Open icon to open the cloned virtual system pattern in the Pattern Builder. Edit the virtual system pattern as needed. You can add or remove images, and then you can add or remove software components, script packages, add-ons, and policies. You can also add or update property mappings between components.
7. When you are finished editing the pattern, click Save.

What to do next

You can lock the virtual system pattern against future editing. For more information about making virtual system patterns read-only, see the related links. You can also deploy the virtual system pattern to the cloud. For more information about deploying, see the related links.

Exporting virtual system patterns

You can export virtual system patterns from the system catalog so that you can use them on a different system.

Before you begin

You must have access to patterns, or access to create patterns, to complete this task.

About this task

You can export a virtual system pattern so that you can use it on a different system. When you export a pattern, the version for the pattern is preserved. You can use the Self-service user interface or the command-line interface to complete
Task. If you use the Self-service user interface to export a pattern, you can export only the pattern model. To export pattern artifacts (script packages, add-ons, images, or software packages) along with the pattern, use the command-line interface instead. For information about using the command-line interface, see the related links.

**Procedure**
1. Click PATTERNS > Pattern Design > Virtual System Patterns.
2. Click Export on the toolbar.
3. Click Save to save the compressed file for the pattern to a local directory.

### Deleting virtual system patterns

You can delete virtual system patterns that you no longer want to deploy.

**Before you begin**

You must have access to the topology to complete this task.

**About this task**

You can use the Self-service user interface, the command-line interface, or the REST API to complete this task. For the command-line and REST API information, see the related links.

**Procedure**
1. Click PATTERNS > Pattern Design > Virtual System Patterns.
2. If the pattern has multiple versions, expand the entry for the pattern.
3. Click Delete in the Actions column for the pattern that you want to delete or select the virtual system pattern and click Delete on the virtual system pattern details page.
4. Click Confirm to confirm that you want to delete the pattern.

### Modifying virtual system patterns

You can modify any virtual system pattern that is not read-only.

**Before you begin**

You must be granted access to the virtual system pattern to complete this task.

**About this task**

Use the Pattern Builder in the Self-service user interface to edit your virtual system pattern. You can add, edit or remove components, links, and policies.

**Procedure**
1. Click PATTERNS > Pattern Design > Virtual System Patterns.
2. Select the virtual system pattern that you want to modify.

    **Note:** If the pattern has multiple versions, expand the entry for the pattern, and then select the version that you want to modify.
3. Click Open on the toolbar.
4. Modify the pattern, as needed. For more information about the assets that are available, see the "Creating virtual system patterns" topic in the related links.

If you want to preserve the previous configuration for the pattern, you can save the modified pattern as a different version by changing the value in the Version field. You also have the option to set a different version when you save the pattern.

5. Save the pattern:
   - Click the Save icon to save the pattern with the same name and version. This option overwrites the existing pattern with the specified name and version.
   - Click Save as to change the name or version for the pattern. This option preserves the original pattern, and saves the modified pattern with the new name and version. If you change only the version, the pattern is displayed by name, and you can expand the entry to see all versions of the pattern.

### Making virtual system patterns read-only

You can deploy both draft and read-only virtual system patterns for testing or production. Making a virtual system pattern read-only prevents further edits to the topology definition and ensures consistent reuse in the cloud.

**Before you begin**

You must have edit access to the virtual system pattern that you want to make read-only. If the pattern is read-only already, you cannot change it. To make a virtual system pattern read-only, ensure that no one with the permission to edit the virtual system pattern intends to change it.

**About this task**

You can make virtual system patterns read-only to prevent further editing of the virtual system pattern.

**Note:** When you make a virtual system pattern read-only, it is possible for the pattern to contain script packages, add-ons, or images that are still in draft state (that is, unlocked). The pattern model is unchanged, but it is still possible to modify the script package, add-on, or image if it is not also locked.

You can use the Self-service user interface, the command-line interface, or the REST API to complete this task. For the command-line and REST API information, see the related links.

**Procedure**

1. Click PATTERNS > Pattern Design > Virtual System Patterns. Virtual system patterns that have a status of Read-only cannot be edited. Virtual system patterns that have a status of Draft can be edited.
2. Select the virtual system pattern that you want to make read-only to show the pattern details.
3. To lock editing of the virtual system pattern, click the Lock icon in the toolbar of the Pattern details page.
   
   You are prompted with a list of any unlocked components in the pattern that can be modified even if the pattern is read-only.
4. Click Confirm to lock the virtual system pattern or Cancel to discard the changes.
CAUTION:
You cannot unlock a virtual system pattern after you lock it.

Results

If you clicked Confirm to lock the virtual system pattern, the pattern is now read-only.

Note: If the pattern contains one or more script packages or add-ons that are not locked, the Current status field in the Virtual System Patterns details page is displayed with the following status:
Read-only (The pattern content can still be modified because some script packages, add-ons, or images are not read-only)

You can lock your script packages and add-ons as needed before you deploy the virtual system pattern. To lock a script package or add-on, return to the appropriate catalog page in the console, select the script package or add-on, and use the Lock function to change the state to read-only.

Deploying virtual system patterns

After you create a virtual system pattern, you can provision and deploy it to the cloud. You can deploy a virtual system pattern multiple times and each deployment is a running virtual system instance on the cloud infrastructure.

Before you begin

Configure the virtual system pattern, and ensure that it is ready to be deployed. For more information about modifying virtual system patterns, see the related links.

Script packages that are included in your virtual system pattern can contain license agreements that you must accept before you can deploy the pattern. For more information about accepting license agreements, see the related links.

To deploy patterns to an externally managed environment profile, all of the components in your pattern must use IBM Foundation Pattern version 2.1.0.0 or later, whether for single-system or multisystem deployment, and must not contain a Hypervisor Edition virtual image.

About this task

You can deploy either draft or committed virtual system patterns for testing or production. The time that it takes to deploy a virtual system depends on several factors, such as the size of the virtual system pattern parts and the interdependencies of parts in the pattern definition, network usage, storage usage, and the provisioning speed of the virtual machine on the cloud infrastructure.

Note: Connectivity issues with the DNS server can cause increased deployment times or failed deployments. The network administrator for the target network must check the routing tables of the DNS server to ensure that it can resolve the network address of the system.

Procedure

1. Click PATTERNS > Pattern Design > Virtual System Patterns.
2. Click **Deploy** in the **Actions** column for the pattern that you want to deploy, or select the pattern that you want to deploy and click **Deploy** on the toolbar.

On the **Configure** pane:

3. Edit the name for the deployment, if needed. This name displays on the Instances page after the pattern deploys.

4. Select the environment profile that you want to use for the deployment.
   - If the network for the selected environment profile is set to **Internally Managed**:
     - Select a **Cloud Group** and an **IP Group**.
     
     **Note:** You can also set the IP group for each virtual machine on the Distribute pane. You cannot change the **Cloud Group** for the deployment after you configure it on this pane.
     
   - The deployment is limited to a single cloud group.
   - If the network for the selected environment profile is set to **Externally Managed**, you select the cloud group and IP group for the deployment later, on the Distribute pane.

5. Set the priority for the deployment.

   **Note:** For more information about deployment priorities, see the Related tasks.

6. Modify the deployment schedule as needed:
   - Choose **Start now**, or choose **Start later** and select a date and time for the deployment to start.
   - Choose **Run indefinitely**, or choose **Run until** and select a date and time for the deployment to end.

7. Optional: To set up SSH access, use one of the following options in the **SSH Key** section to set the public key:
   - To generate a key automatically, click Generate. Click Download to save the private key file to a secure location. The default name is `id_rsa.txt`. The system does not keep a copy of the private key. If you do not download the private key, you cannot access the virtual machine, unless you generate a new key pair. You can also copy and paste the public key into a text file to save the key. Then, you can reuse the same key pair for another deployment. When you have the private key, make sure that it has the correct permissions (`chmod 0400 id_rsa.txt`). By default, the SSH client does not use a private key file that provides open permission for all users.
   - To use an existing SSH public key, open the public key file in a text editor and copy and paste it into the **SSH Key** field.

   **Important:** Do not use `cat`, `less`, or `more` to copy and paste from a command shell. The copy and paste operation adds spaces to the key that prevent you from accessing the virtual machine.

   The SSH key provides access to the virtual machines in the cloud group for troubleshooting and maintenance purposes. See the topic, "Configuring SSH key-based access", for details about SSH key-based access to virtual machines.

8. Modify the pattern and component attributes as needed.

   The attributes that display in the pattern configuration column are attributes from the pattern and components in the pattern that are not locked from editing. You can modify existing values or set values that were not specified during pattern creation. Be sure that all required fields have values.
Components that have a blue dot next to the name contain required attributes that must be set before the pattern is deployed.

9. When you are finished configuring all of the fields on the Configure tab, click **Continue to distribute**.

On the Distribute pane:

The virtual machines in the deployment are placed in cloud groups by the system.

10. Optional: To modify the placement of the virtual machines, drag the virtual machines to different cloud groups.

   • If you drag a virtual machine cell that contains more than one virtual machine, you are prompted to select the number of virtual machines that you want to move. You must select the number from the list in the dialog. After you move a virtual machine to a different cell, the IP group assignments are set to default values. If needed, you can edit the virtual machine network settings in the next step to modify the IP group.

   • If you modify the placement of the virtual machines, the new placement is validated to ensure that the necessary resources and artifacts are available in the selected cloud group.

   • If there is a problem with the placement, a message is displayed. Resolve the issue with the placement before you continue.

   For example, if this message displays when you modify the placement: CWZKS700Z0E Insufficient memory to place the pattern, move the virtual machine to a different cloud group with sufficient memory resources for the pattern.

   If you see the error: Unable to assign to cloud group, there is an error with the location, cloud group, NIC or IP groups for the cell where the error is displayed. If this error message occurs, you must resolve the issue with that cell before you are allowed to drag a virtual machine to that cell for placement there. Hover your mouse over the error to display more details about the problem in a pop-up window.

11. To edit the network or storage volume settings for a virtual machine, hover your mouse over the virtual machine icon and click the pencil icon.

   a. On the IP Groups tab, you can modify IP group for each of the NICs in the virtual machine. The IP groups that are listed are associated with the environment profile that you chose for the deployment. If the IP address provided by field in the environment profile that you chose for the deployment is set to Pattern Deployer, you must set the IP address for each NIC in the deployment.

   b. If there is a Default attach block disk add-on in the pattern, you can modify the storage volumes for the virtual machine on the Storage Volumes tab. You can use an existing storage volume or create one to attach to the component during deployment. If you choose to create a new storage volume, configure these settings:

      Name Set the name for the storage volume.
      Description Optional. Set a description for the storage volume.
      Size (GB) Set the size for the storage volume, in GB.
Volume Groups
Select a volume group for the storage volume. A storage volume group is a logical grouping of volumes that can span workloads and cloud groups.

c. Click OK when you are finished updating the settings.

12. When you are finished modifying the settings, click Deploy.

When the virtual system is deployed, the virtual system instance is listed under the Virtual System Instances section of the IBM Cloud Orchestrator Self-service user interface. To view the virtual system instance, click PATTERNS > Instances > Virtual System Instances.

The virtual memory and virtual processor settings that are configured for the virtual images in the virtual system pattern must be met by the requirements for the software components in the pattern. If these requirements are not met, the deployment fails and an error message that lists the memory and processor requirements is displayed. If this error occurs, modify the processor and memory settings in the pattern so that the requirements are met, and deploy the pattern again.

13. View the details of the virtual system instance in the Virtual System Instances page.

Results

The placement is validated again to ensure that the resources and artifacts that were used for validation during the initial placement are still available. If there is a problem with the placement, an error message is displayed, and a red circle is displayed on the circle that contains errors. Hover over the cell that contains errors, and then hover over the yellow triangle in the resulting pop-up window to view more details about the errors. Resolve the issue with the placement, such as moving the virtual machine to a different system with sufficient resources so that the deployment can continue.

After placement validation is successful, the virtual system instance is deployed and started. To stop the virtual system instance, select the virtual system instance from the list, and click Stop. To start the virtual system instance again, select the virtual application click Start.

To remove a stopped topology, select it from the Virtual System Patterns page, and click Delete.

What to do next

After you deploy the virtual system instance, you can use the IP address of the virtual machines to access the application artifacts. For example, you can manually enter the URL in your browser.
http://IP_address:9080/tradelite/

IP_address is the IP address of the deployed WebSphere Application Server virtual machine.

If you uploaded an SSH public key during the deployment, you can also connect directly to a virtual machine without a password if you have the private key.

You can also view and monitor statistics for your deployed virtual machines and download and view the log files. For more information, see the Related tasks section.
Working with virtual system patterns (classic)

Using a virtual system pattern, you can describe the topology of a system that you want to deploy. Virtual system patterns provide repeatable system deployment that can be reproduced. To build virtual system patterns, you can use parts from one or more virtual images, add-ons, and script packages.

About this task

To work with virtual system patterns (classic), perform the following steps:

Procedure

1. Use the user interface, as described in "Working with virtual system patterns (classic) in the user interface" on page 412.
2. Configure the advanced options. You can configure advanced options for the virtual system patterns, as described in “Configuring advanced options” on page 425.

Supported virtual system patterns (classic)

You can use predefined virtual system patterns that are provided by IBM, or you can clone them and customize the copies to create new virtual system patterns that are more suitable to your environment.

Patterns can be downloaded from the IBM PureSystems Centre.

For example, you can download the following virtual system patterns:

- DB2 Enterprise Server Edition
- IBM WebSphere Application Server Hypervisor Edition

To start working with a virtual system pattern, choose any of the following options:

- Use a predefined virtual system pattern.
  If one of the predefined virtual system patterns meets the needs of your environment, you can use it without altering it and deploy it to the cloud. For details, refer to “Using a predefined virtual system pattern (classic)” on page 413.

- Clone an existing virtual system pattern.
  If one of the predefined virtual system patterns closely meets your needs but you must customize it, you can clone the virtual system pattern and then modify the copy. For details, refer to “Cloning an existing virtual system pattern (classic)” on page 414.

- Create a virtual system pattern.
  If the predefined or cloned virtual system patterns do not meet the needs of your environment, you can create a virtual system pattern. For details, refer to “Creating a virtual system pattern (classic)” on page 416.
Working with virtual system patterns (classic) in the user interface

Virtual system patterns implement deployment topologies from one or more virtual images and applications from the IBM Cloud Orchestrator catalog.

Before you begin

Review the available virtual system patterns, including the virtual system patterns provided by IBM. You can then determine if an existing virtual system pattern is suitable to your environment or if there is one that you can customize to be suitable. See “Supported virtual system patterns (classic)” on page 411 for information about the provided virtual system patterns.

About this task

You can use the virtual system patterns provided by IBM, as they are, and deploy them to your cloud. You can also use the IBM Cloud Orchestrator virtual system pattern editor to alter the topology and configuration of your environments. In a single virtual system pattern, you can include parts from multiple images and you can drag script packages and add-ons onto any of the parts in the virtual system pattern. You can configure parameters for the parts. If the script packages or add-ons you are using have parameters, you can also configure the parameters.

Information about each of the fields in the Virtual System Patterns (Classic) window is provided in “Virtual system pattern (classic) windows” on page 445.

Procedure

1. From the menu, open the Virtual System Patterns (Classic) window by clicking PATTERNS > Instances > Virtual System Patterns (Classic).
2. Determine the task to perform. You can perform the following tasks with the IBM Cloud Orchestrator user interface:
   - **Determine the virtual system pattern to use.**
     You can use a predefined virtual system pattern, clone an existing virtual system pattern, or create a virtual system pattern, as described in “Selecting a virtual system pattern (classic)” on page 413.
   - **Edit a virtual system pattern.**
     You can edit any virtual system pattern that is not read-only and which you have permission to edit, as described in “Editing a virtual system pattern (classic)” on page 418.
   - **Configure advanced options.**
     You can edit and configure the advanced options for a virtual system pattern, as described in “Configuring advanced options” on page 425.
   - **Make a virtual system pattern read-only.**
     If you want to lock a virtual system pattern to future editing, you can make it read-only, as described in “Making virtual system patterns (classic) read-only” on page 439.
   - **Deploy a virtual system pattern.**
     You can deploy the virtual system pattern after you have configured it, as described in “Deploying a virtual system pattern (classic)” on page 440.
Delete a virtual system pattern.
You can delete a virtual system pattern that you do not need any more, as described in “Deleting a virtual system pattern (classic)” on page 444.

Selecting a virtual system pattern (classic)
You can use a set of predefined virtual system patterns that is provided by IBM. You can also clone an existing IBM Cloud Orchestrator virtual system pattern to customize it or create a virtual system pattern.

Before you begin
Review the available virtual system patterns, including the virtual system patterns provided by IBM. Determine if an existing virtual system pattern is suitable to your environment or if there is one that you can customize to be suitable. See “Supported virtual system patterns (classic)” on page 411 for information about the provided virtual system patterns.

About this task
You can use the predefined virtual system patterns provided by IBM, as they are, clone and edit an existing virtual system pattern, or create a virtual system pattern.

Procedure
• Use a predefined virtual system pattern. If one of the predefined virtual system patterns meets the needs of your environment, you can use it without altering it and deploy it to your cloud. See “Using a predefined virtual system pattern (classic)” for more information.
• Clone an existing virtual system pattern. If one of the predefined virtual system patterns closely meets your needs but you must customize it, you can clone the virtual system pattern and then edit the copy. See “Cloning an existing virtual system pattern (classic)” on page 414 for more information. See “Editing a virtual system pattern (classic)” on page 418 for more information.
• Create a virtual system pattern. If the predefined or cloned virtual system patterns do not meet the needs of your environment, you can create a virtual system pattern. See “Creating a virtual system pattern (classic)” on page 416 for more information.

What to do next
When you have completed any necessary work with the virtual system pattern, you can deploy the virtual system pattern to your cloud. See “Deploying a virtual system pattern (classic)” on page 440 for more information.

Using a predefined virtual system pattern (classic):
You can use a set of predefined virtual system patterns that is provided by IBM. Virtual system pattern are made up of parts from one or more virtual images, and script packages from the IBM Cloud Orchestrator catalog. Virtual system patterns provide a topology definition for repeatable deployment that can be shared.

Before you begin
Review the virtual system patterns provided by IBM Cloud Orchestrator to determine which virtual system pattern best fits your needs. See “Supported virtual system patterns (classic)” on page 411 for more information.
virtual system patterns (classic)” on page 411 for information about these virtual system patterns.

About this task

You can use the virtual system patterns provided by IBM Cloud Orchestrator, as they are, and deploy them to your cloud.

Procedure

1. From the list in the left panel select a predefined virtual system pattern.
2. Add it to the cloud. Click the Deploy icon to provide the necessary information to deploy this virtual system pattern.
3. Deploy the virtual system pattern. When all of the information is provided correctly in the Deploy pattern dialog, click OK to deploy the virtual system pattern. A green check mark beside each entry indicates that the information has been provided. For more information about deploying a virtual system pattern, see “Deploying a virtual system pattern (classic)” on page 440.

Results

The virtual system pattern is now running in the virtual system instance.

Cloning an existing virtual system pattern (classic):

IBM provides a set of predefined virtual system pattern that you can clone. Because the predefined virtual system patterns cannot be edited, cloning them provides a starting point for creating customized virtual system patterns that work in your environment.

Before you begin

You must be granted access to the pattern and be assigned the catalogeditor role or the admin role.

Select a virtual system pattern that most closely meets your needs. See “Supported virtual system patterns (classic)” on page 411 for virtual system pattern descriptions.

Because virtual system patterns are associated with virtual images, if you have not accepted the license for the virtual image with which the virtual system patterns are associated, the Clone option is not available. If the clone function is not available for the virtual system pattern you want to use, accept the license for the image associated with the virtual system pattern. To accept the license, click PATTERNS > Pattern Design > Virtual Images. See Chapter 6, “Managing virtual images,” on page 325 for more information about virtual images.

Important: You can accept a license and then change the image that the virtual system pattern is using when you define the cloned virtual system pattern. If you change the image and do not actually use the image for which you accepted the license, you are not charged for that license.

About this task

This task provides the necessary steps to clone a virtual system pattern and then customize the copy to meet the needs of your environment.
Procedure

1. From the left panel Virtual System Patterns (Classic) window, select a virtual system pattern to clone. The description and general information about this virtual system pattern display in the right panel of the Virtual System Patterns (Classic) window.

2. Clone the virtual system pattern. Click the clone icon on the upper right panel of the Virtual System Patterns (Classic) window.

3. You can provide the following basic information about the new virtual system pattern you are cloning:
   - **Name** Enter a unique name for the virtual system pattern in the Name field. This information is required to clone a virtual system pattern.
   - **Description** Optionally, enter a detailed description to identify the virtual system pattern in the Description field.

4. Virtual image
   Select a virtual image with which to associate the virtual system pattern from the listing. This information is required to clone a virtual system pattern. You can edit the new virtual system pattern to associate individual parts with different virtual images. If all the parts in the virtual system pattern you are cloning are from a single virtual image, use this option. This option switches all of the parts to a different virtual image in the new virtual system pattern. If the original virtual system pattern contains parts from different virtual images, this option is disabled. If this option is disabled, the parts in the new virtual system pattern are associated with the same virtual images as the corresponding parts in the original virtual system pattern.

5. Click OK to save your changes. When the information is processed, you return to the Virtual System Patterns (Classic) window and the virtual system pattern you created is added to the list in the left panel. It is selected so that the information about it is shown in the right panel. For more information about the fields on this panel, see “Virtual system pattern (classic) windows” on page 445.

6. Edit the virtual system pattern. To change the virtual system pattern topology, click the edit icon on the upper right panel of the Virtual System Patterns (Classic) window. You can perform the following actions with virtual system patterns:
   - Add or remove parts
   - Edit parts
   - Add or remove script packages to the parts
   - Add or remove add-ons to the parts
   - Configure properties for the parts and parameters for the script packages that have parameters
   - Define advanced options
   - Modify the start up order of the parts

   The Pattern Editor window provides a list of parts. For more information about the interaction of the parts on the Pattern Editor window, see “Virtual system pattern (classic) editing views and parts” on page 419.

6. Edit the parts on the canvas. See “Editing a virtual system pattern (classic)” on page 418 for more information about editing functions you can perform on virtual system patterns.
7. Edit advanced options. Default advanced options are provided with the virtual system patterns but you can edit those settings. For more information, see “Configuring advanced options” on page 425.

8. Complete the virtual system pattern. When you have finished editing this virtual system pattern, click the Done editing link on the upper right panel of the Pattern Editor window. This virtual system pattern is listed on the left panel of the Virtual System Patterns (Classic) window.

Results

When you have completed these steps, you have cloned the virtual system pattern and it can be customized.

What to do next

You can lock the virtual system pattern against future editing. For more information, see “Making virtual system patterns (classic) read-only” on page 439.

You can deploy the virtual system pattern to the cloud. For more information, see “Deploying a virtual system pattern (classic)” on page 440.

Creating a virtual system pattern (classic):

You can create a virtual system pattern using the IBM Cloud Orchestrator user interface. Virtual system patterns are topology definitions for repeatable deployment that can be shared.

Before you begin

You must be assigned the catalogeditor role or the admin role.

Review the predefined virtual system patterns to ensure that none of the existing virtual system patterns can be cloned and customized to meet your needs. For more information about the predefined virtual system patterns, see “Supported virtual system patterns (classic)” on page 411.

About this task

You can create a virtual system pattern by cloning an existing virtual system pattern or by creating a virtual system pattern. This task provides the steps for creating a virtual system pattern.

Procedure

1. Add a virtual system pattern. On the upper left panel of the Virtual System Patterns (Classic) window, click Add to provide the following basic information about the virtual system pattern you are creating.

   - **Name**: Enter a unique name for the virtual system pattern in the Name field. This information is required to create a virtual system pattern.

   - **Description**: Optionally, enter a detailed description to identify the virtual system pattern in the Description field.

2. Click OK to indicate that you have finished editing and return to the initial view of the virtual system pattern. When the information is processed, you return to the Virtual System Patterns (Classic) window and the virtual system pattern you created is added to the list in the left panel. It is selected so that
the information about it is shown in the right panel. For more information about the fields on this panel, see "Virtual system pattern (classic) windows" on page 445.

3. Edit the virtual system pattern. To change the virtual system pattern topology, click edit on the upper right panel of the Virtual System Patterns (Classic) window. You can perform the following actions:
   - Add or remove parts
   - Edit parts
   - Add script packages to the parts
   - Add or remove add-ons to the parts
   - Remove script packages from the parts
   - Configure properties for the parts
   - Configure parameters for script packages that have them
   - Define advanced options
   - Modify the start up order of the parts

   The Pattern Editor window provides a list of parts. For more information about the interaction of the parts on the Pattern Editor window, see "Virtual system pattern (classic) editing views and parts" on page 419.

4. Edit the parts on the canvas. See "Editing a virtual system pattern (classic)" on page 418 for more information about editing functions you can perform on virtual system patterns.

5. Edit advanced options. Default advanced options are provided with the virtual system patterns but you can edit those settings. For more information, see "Configuring advanced options" on page 425.

6. Optional: Modify the default order in which the parts run at deployment. See "Ordering parts to run at deployment" on page 422 for more information.

7. Indicate that you have finished editing and return to the initial view of the virtual system pattern. When you have finished editing this virtual system pattern, click the Done editing link on the top of the right panel of the Pattern Editor window.

Results

When you have completed these steps, you have configured basic information about the virtual system pattern you have created and it can be deployed to the cloud.

What to do next

You can lock the virtual system pattern against future editing. For more information, see "Making virtual system patterns (classic) read-only" on page 439.

You can deploy the virtual system pattern to the cloud. For more information, see "Deploying a virtual system pattern (classic)" on page 440.
Editing a virtual system pattern (classic)

You can edit any virtual system pattern that is not read-only. You can modify a virtual system pattern to suit the changing needs of your environment.

Before you begin

You can edit virtual system patterns that are not read-only and locked to editing. If a virtual system pattern can be edited, the edit icon is shown beside it. If a virtual system pattern is locked to editing, the locked icon is shown beside it.

To modify an existing virtual system pattern that is not read-only, you must have either created the virtual system pattern or have been given the correct user access. The owner for each virtual system pattern is shown on the Virtual System Patterns (Classic) window for that virtual system pattern. If you do not have write access to the virtual system pattern you want to edit, you can request access from that owner.

About this task

You can edit any virtual system patterns that are not read only and to which you have write access. This task provides information about editing existing virtual system patterns.

Procedure

1. Select the virtual system pattern you want to edit from the left panel of the Virtual System Patterns (Classic) window. Details about the virtual system pattern are shown in the right panel of the Virtual System Patterns (Classic) window.
2. From the top of the right panel of the Virtual System Patterns (Classic) window, click the edit icon The Virtual System Patterns (Classic) window opens for this virtual system pattern.
3. Edit the parts on the canvas.
   a. Select a part from the lists on the left panel of the Pattern Editor window. The lists of parts, script packages, and add-ons show available parts that can be dropped onto the editing canvas on the right side of the Pattern Editor window.
   b. Drop the selected parts onto the editing canvas on the right of the Pattern Editor window.

   The editing canvas graphically shows the topology of the virtual system pattern. See “Virtual system pattern (classic) editing views and parts” on page 419 for information about the virtual image parts and the interaction between them.
4. Optional: Configure advanced options. For more information, see “Configuring advanced options” on page 425.
5. Optional: Configure the order in which the parts are to deploy. For more information, see “Ordering parts to run at deployment” on page 422.
6. Indicate that you have finished editing and return to the initial view of the virtual system pattern. When you have finished editing this virtual system pattern, click the Done editing link on the top of the right panel of the Virtual System Patterns (Classic) window.
Results

When you have finished editing this virtual system pattern, it is ready to be deployed to the cloud.

What to do next

You can lock the virtual system pattern against future editing. For more information, see “Making virtual system patterns (classic) read-only” on page 439. You can deploy the virtual system pattern to the cloud. For more information, see “Deploying a virtual system pattern (classic)” on page 440.

Virtual system pattern (classic) editing views and parts:

A virtual system pattern, that is not read-only, can be edited if you have permission to edit it. The topology for a virtual system pattern is graphically shown. Virtual image parts, add-ons, and script packages can be dropped onto an editing canvas to create or change relationships between the parts that define the topology.

The Virtual System Patterns (Classic) window

When you select a virtual system pattern to edit in Virtual System Patterns (Classic) window, you can see information about the virtual system pattern. The topology of the virtual system pattern is shown on the right panel of the Virtual System Patterns (Classic) window. For more information about the predefined virtual system patterns and what they provide, see “Supported virtual system patterns (classic)” on page 411.

The Pattern Editor window

Clicking the edit icon on the upper right panel of the Virtual System Patterns (Classic) window opens the Pattern Editor window for the selected virtual system pattern. The Pattern Editor window provides lists to select virtual image parts, add-ons, and script packages.

Virtual image parts

Selecting the Parts list on the Pattern Editor provides a listing of the parts that can be dropped onto the virtual system pattern canvas. The virtual system pattern canvas is on the right panel of the Virtual System Patterns (Classic) window. The following virtual image parts are examples of the parts available for IBM WebSphere Application Server Hypervisor Edition images:

- Administrative agents
- Custom nodes
- Deployment manager
- IBM HTTP servers
- Job manager
- Stand-alone server
- On-demand router: The on-demand router part is available if you are using the WebSphere Application Server 7.0.0.17 with Intelligent Management Pack image.

These parts are determined by the virtual images you are using. For more information about virtual images, see Chapter 6, “Managing virtual images,” on page 325.
Some virtual image parts represent multiple instances. These graphical parts on the editing canvas have a badge that shows the number of instances of the part. A valid number of instances that can be specified is 1 - 999.

You can configure the parts either when you deploy the virtual system pattern or directly from the part before deployment. To configure the part before deploying it, click the edit properties icon for the part on the editing canvas. For more information about configuring the parts, see “Configuring parts” on page 423.

Script packages

The Scripts list on the Pattern Editor provides a listing of the script package parts that can be dropped into the virtual image parts. Virtual image parts are on the right panel of the Virtual System Patterns (Classic) window. This list can contain script packages associated with the virtual image and any that you have defined for use with IBM Cloud Orchestrator. For more information about script packages, see “Adding a script package” on page 357 and “Associating a script package with a pattern” on page 361. Script packages can then be added to the virtual image parts.

Add-ons

The following default add-ons are provided with IBM Cloud Orchestrator and can be added to parts on the editing canvas:

Default add NIC

Adds a new virtual network interface controller (NIC) to the virtual machine, configures its IP address information, and activates it. Use this add-on for virtual image parts that support communication using ssh to run scripts after initial activation of the virtual machine.

Note: This add-on is not supported on PowerVM virtual images.

Default configure NIC

Triggers configuration via VSAE of the additional NICs present in the image.

Note: This add-on is not supported on PowerVM virtual images.

Default add disk

Adds a virtual disk to the virtual machine and optionally formats and mounts the disk. Prerequisite: the parted (parted RPM) and sfdisk (util-linux RPM) tools must be installed for a RedHat image (or other type of packages depending on different operating systems). The prerequisite for VMware virtual images is that the virtual disk type must be SCSI.

Default AIX add disk

Adds a virtual disk to the virtual machine and optionally formats and mounts the disk. Use this add-on for PowerVM virtual images.

Note: IBM Cloud Orchestrator does not support the disk add-on function for PowerVM with Shared Storage pool.

Default Windows add disk

Adds a virtual disk to the virtual machine and formats and makes the disk available under new letter. Prerequisites: PowerShell and Diskpart tools, which are available in the default Windows installation. New disk is
partitioned using MBR schema. The prerequisite for VMware virtual images is that the virtual disk type must be SCSI.

**Default raw disk**

Adds a virtual disk to the virtual machine, the disk is added raw without partitions or formatting. For PowerVM virtual images, you must run the `cfgmgr` to refresh the system configuration and to see the new disk. The prerequisite for VMware virtual images is that the virtual disk type must be SCSI.

**Note:** IBM Cloud Orchestrator does not support the disk add-on function for PowerVM with Shared Storage pool.

**Default add user**

Defines an additional user on the virtual machine. The default add-on script runs a simple add user command. No additional account configuration is performed.

Customized versions of each of these add-on types can also be available if a user has cloned or created new add-ons in the catalog. For more information about managing add-ons in the catalog, see "Managing add-ons" on page 385. Add-ons are run before script packages.

**Parts on the editing canvas**

When you drop the parts onto the canvas on the right panel of the Virtual System Patterns (Classic) window, they interact in specific and predictable ways. Though there are no columns visible and there are no column labels on the canvas, objects fall into general groups. The objects are placed in the following general locations on the canvas:

**Managers on the left**

The left side of the editing canvas contains parts that act as managers managing other parts. For example, this column contains deployment managers and job managers. These managers manage the objects (nodes and connections) in the other two sections. You can add a manager and then add the nodes and the nodes federate to the manager. Or, you can add the nodes and then add a single manager and the nodes federate to the manager. You can have both a deployment manager and a job manager in one virtual system pattern.

**Nodes in the center**

The center area contains managed nodes, for example custom nodes, administrative agents or stand-alone nodes. The nodes are automatically federated into the managers in the left column. Administrative agents are registered with the job manager. Stand-alone servers can be federated to a deployment manager if one is present.

**Connections on the right**

The right area contains connection parts. For example, the right of the editing canvas can contain IBM HTTP Server parts that route the traffic for the nodes. The right area can also contain an On demand routers part if you are editing a virtual system pattern from an IBM WebSphere Application Server Hypervisor Edition Intelligent Management Pack virtual image.
Interaction between virtual image parts

Virtual image parts can be defined to interact with other virtual image parts. When the interacting virtual image parts are included in the same virtual system pattern, cross configuration results. For example, when a custom node (or IBM HTTP Server) and a deployment manager are placed in the same virtual system pattern, they are automatically cross configured. This results in the custom node being federated to the deployment manager. Similarly, administrative agents (or deployment managers) are registered with a job manager.

Virtual image parts can be cross configured if the virtual system pattern editor can determine a unique relationship. If it is unable to do so, no cross configuration occurs. For example, if a custom node is added to a virtual system pattern with two deployment managers, no federation takes place. However, if one of the deployment managers is later removed, cross configuration occurs because a unique relationship now exists.

If you have enabled the WebSphere Application Server 7.0.0.17 with Intelligent Management Pack or later image, an on-demand router part is available to use in your virtual system patterns. If the on-demand router part is used in a custom virtual system pattern, a deployment manager part that is enabled with Intelligent Management Pack is also required. Without the deployment manager part that is enabled for the Intelligent Management Pack, virtual system patterns with on-demand router part are not valid. However, if there is a deployment manager part that is enabled with Intelligent Management Pack, a custom node part that is enabled with Intelligent Management Pack is not required.

You can use the version indicator on the parts to ensure that they are referencing the same version of the virtual image in the catalog. For example, the deployment manager part and the on-demand router part must reference a virtual image version that is enabled with Intelligent Management Pack. These two parts reference the same version of the virtual image in the catalog. If the version of a part is incorrect, you can change it when the part is on the editing canvas. Hovering over the part name opens a window with additional information about the virtual image.

Ordering parts to run at deployment:

When you create a virtual system pattern, parts, scripts and add-ons run in a specific order at deployment. Add-ons always run before scripts, for example. You can change the order in which parts and scripts run with the user interface.

Before you begin

To modify the order of the parts that are deployed for an existing virtual system pattern (that is not read-only), you must have either created the virtual system pattern or have been given the correct user access. The owner for each virtual system pattern is shown on the Virtual System Patterns (Classic) window for that virtual system pattern. If you do not have write access to the virtual system pattern you want to edit, you can request access from that owner.

Note: The order of default parts cannot be edited when they are grayed out. The ordering on these parts is the default ordering provided by a virtual system pattern template.
About this task

You can change the order in which parts, and some scripts, run when a virtual system pattern is deployed. You can order any scripts that run when the virtual machine is created. You cannot order any scripts that are initiated by a user and you cannot order deletion script packages.

Procedure
1. Select the virtual system pattern you want to edit from the left panel of the Virtual System Patterns (Classic) window.
2. From the upper right of the Virtual System Patterns (Classic) window, click the edit icon. The Pattern Editor window is opened for the selected virtual system pattern.
3. Click Ordering from the upper right of the Pattern Editor window. The right editing panel of the Pattern Editor window displays the parts by category: part or script.
4. Change the order of the parts. You can order the parts to deploy by group. Groups of parts are numbered and labelled with the clock icon. Dragging the parts adds constraints. Parts can be moved between groups or to create new groups. Parts within a group do not necessarily deploy in the order in which they are shown within the group. The groups do deploy relative to the other groups above and below them. The text to the left of the part and script listing describes the order of deployment for each part.
5. Indicate that you have finished editing and click Topology to return to the initial view of the virtual system pattern. When you have finished editing this virtual system pattern, click the Done editing link on the top of the right panel of the Pattern Editor window.

Results

When you have finished editing this virtual system pattern, it is ready to be deployed to the cloud.

What to do next

You can configure the advanced options, or deploy the virtual system pattern to the cloud. For more information, see "Configuring advanced options" on page 425 and "Deploying a virtual system pattern (classic)" on page 440.

Configuring parts
Before deploying a virtual system pattern to run in a cloud group, you must first configure the parts included in the virtual system pattern.

Before you begin

You can configure the parts included in your virtual system pattern in the following ways:
- You can configure the part when you deploy the virtual system pattern.
- You can configure the part properties from the editing canvas of the Pattern Editor window.

For more information about modifying the pattern or configuring advanced options, see the related links.
**About this task**

To configure parts for a virtual system pattern, the particular virtual system pattern is either open and being edited or it is being deployed. When editing part properties from the virtual system pattern, you can lock the values so that they cannot be changed during deployment. The parts that require information are different depending on the type of virtual image to be deployed and the type of hypervisors in the cloud. For example, parts for an WebSphere Application Server image would require different configuration than parts for a DB2 image.

**Procedure**

1. Open the **Properties** configuration panel for the part by using one of the following methods:

   **Editing the virtual system pattern**
   From the Virtual System Patterns (Classic) window, select the virtual system pattern to edit and click the **Edit** icon. For each virtual image part requiring information, click the **Properties** icon on the part. You can also configure any script packages or disk or user add-ons on the parts. NIC add-ons require an environment profile for configuration.

   **Deploying the virtual system pattern**
   To deploy a virtual system pattern from the Virtual System Patterns (Classic) window, click the **Deploy** icon. on the upper right of the Virtual System Patterns (Classic) window. When deploying a virtual system pattern, you must describe the virtual system instance that you want to deploy. As part of that process, the parts in the virtual system pattern to deploy are listed.

   When the information for each of the virtual image parts in your virtual system pattern is provided, a green check mark is shown to the left of the virtual image part. If information for one of these parts is missing, then the check box to the left of the **Configure virtual parts** field does not contain a green check mark. In this case, click **Configure virtual parts** and then click the link for the virtual image part that is missing information.

2. Provide the necessary information. Part properties vary, depending on the type of part you are editing, and the scripts and add-ons it includes. If the script packages have parameters, you can also edit these properties.

   You can edit the following properties for add-ons while editing the part or during the deployment process:

   **Disk add-on**
   Has the following properties to edit:
   - DISK_SIZE_GB
   - FILESYSTEM_TYPE
   - MOUNT_POINT

   **Raw disk add-on**
   Has the following properties to edit:
   - DISK_SIZE_GB
   - FILESYSTEM_TYPE

   **User add-on**
   Has the following properties to edit:

   **Note:** The FILESYSTEM_TYPE property is read-only.
• USERNAME
• PASSWORD
• Verify password

Note: NIC add-ons require an environment profile for configuration.

3. Optional: Lock the properties. If you are editing part properties from the virtual system pattern, you can lock the values so that they cannot be changed during deployment. Use the unlocked or a locked icons next to each field on the Properties window to change the status of the field. By default, the part properties are not locked so you must lock them if you want to prevent them from being changed during deployment.

Results

The virtual image parts for the virtual system pattern are configured.

What to do next

Deploy the virtual system pattern to the cloud.

Configuring advanced options

When you have edited the topology of a virtual system pattern, you can configure advanced function for the virtual system pattern.

Before you begin

Ensure that you have access to edit the virtual system pattern you want to work with and that it is not read-only. You can configure advanced options for virtual system patterns you have created or virtual system patterns you have access to edit.

About this task

When you have created or edited the initial topology for a virtual system pattern, there are additional advanced options that you can configure. Advanced options include messaging, session persistence, and global security. The options that are available depend on the topology of the virtual system pattern you are editing. The predefined virtual system patterns provided by IBM are of two basic types of topologies: single server virtual system patterns and cluster virtual system patterns.

Important: It may happen that advanced options are not available at all. It depends on the items that were used in the topology of the virtual system pattern you created.

Procedure

1. From the left panel of the Virtual System Patterns (Classic) window, select the virtual system pattern.
2. Put the virtual system pattern in edit mode. Click the Edit icon at the top of the right panel to see the topology of the virtual system pattern and edit it.
3. Edit the advanced options. Click the Advanced Options... link on the right panel of the Pattern Editor window. The options available on this panel depend on the topology of the virtual system pattern you are editing.
Important: When you open the advanced options editor for a new virtual system pattern that has no advanced options set, default settings are shown for the virtual system pattern. These settings are recommended values for the topology. To accept these default values for this topology, click OK. To return to the virtual system pattern without setting these default values, click Cancel.

The following general options are available:

**Single server virtual system patterns**
- Enable session persistence
- Global security

For detailed information about advanced configuration options for single server virtual system patterns, see [“Configuring advanced options for single server virtual system patterns (classic)” on page 437](#).

**Cluster virtual system patterns**
- Define clusters
  - Enable messaging
  - Enable session persistence
  - Global security

For more information about the advanced configuration options for cluster virtual system patterns, see [“Configuring advanced options for cluster virtual system patterns (classic)” on page 427](#).

**IBM WebSphere Application Server Hypervisor Edition Intelligent Management Pack cluster virtual system patterns**
If the cluster virtual system pattern you are editing is from an Intelligent Management Pack image, then the following options are also available:
- Define dynamic clusters
- Enable overload protection
- Configure standard health policies
- On demand router-dependent health policies

For more information about the advanced configuration options for Intelligent Management Pack cluster virtual system patterns, see [“Configuring advanced options for Intelligent Management Pack” on page 431](#).

4. Save your changes. When you change the settings and click OK, your changes are saved in place of the default values.

5. Optional: Configure advanced messaging options for cluster virtual system patterns. If you are working with a cluster virtual system pattern and you want to configure advanced messaging for it, see [“Configuring advanced messaging for databases” on page 436](#).

6. Optional: Enable the database implemented session persistence option. If you are enabling session persistence for either a cluster or single server virtual system pattern, you must enable the database implemented session persistence. See [“Configuring database implemented session persistence for Derby” on page 439](#) for more information.

**What to do next**

You can perform the following tasks after configuring the advanced options for a virtual system pattern:
Configure the parts
For more information about configuring the parts in your virtual system pattern, see “Configuring parts” on page 423.

Make the virtual system pattern read only
For more information about locking the virtual system pattern against future editing, see “Making virtual system patterns (classic) read-only” on page 439.

Deploy the virtual system pattern
For more information about deploying the virtual system pattern to a cloud, see “Deploying a virtual system pattern (classic)” on page 440.

Configuring advanced options for cluster virtual system patterns (classic):

You can use the advanced virtual system pattern configuration function as a starting point in configuring the cluster virtual system pattern that defines your cell.

Before you begin

Configure the parts and topology for your cluster virtual system pattern before configuring the advanced options. When you open the advanced options editor for a new cluster virtual system pattern, default settings are shown for the virtual system pattern. These settings are typical default values for the topology. To accept these default values for this topology, click OK. To return to the virtual system pattern without setting these default values, click Cancel.

About this task

The following options are available when you are editing a cluster virtual system pattern:

• Define clusters
  – Enable messaging
  – Enable session persistence
  – Global security

Procedure

1. Define clusters. Use this option to configure the number of clusters and application servers on the deployment manager part. You can configure messaging, session persistence, and global security. Using the define clusters option provides the following features, all of which you can change:
  • An application cluster with the default name prefix HVWebCluster
  • A default number of clusters
  • A default number of servers per node

   The server names are in the prefix + cluster index + node name + server index format, as shown in the following example:
   \[HVWebCluster_1_{myNode}_1\]

2. Enable messaging. Use this function to configure additional message engine configuration on the deployment manager part. Messaging engines point to WebSphere Derby on the deployment manager. On the virtual system instance, start Derby or configure the deployment manager to use another database.
Important: You must have the **Define clusters** option selected to work with messaging.

For more information about advanced configuration for WebSphere Application Server clustered messaging engines or a sample authentication alias for databases, see [“Configuring advanced messaging for databases” on page 436](#).

Use the following options to configure messaging with IBM Cloud Orchestrator:

**Standard messaging engine configuration**

When configuring the standard Java Message Service (JMS), IBM Cloud Orchestrator provides the following function:

- An application cluster with the default name prefix HVMsgCluster (which you can change)
- A default number of clusters (which you can change)
- A default number of servers per node (which you can change)
- A Service Integration Bus (SIBus), the name of which has the HVSIBUS prefix, is created for each message cluster
- A messaging engine (ME) is defined on each member of the cluster, because each message cluster is added to the SIBus
- A Derby Java Database Connectivity (JDBC) provider and Derby data source are defined for use by the messaging engine or engines defined. See [“Configuring advanced messaging for databases” on page 436](#) for more information.
- An example authentication alias provides configuration options for the messaging engine to a database other than Derby
- Activation in only one of the members as the messaging cluster members are started

**Highly available messaging engine configuration**

Processed over standard Java Message Service (JMS) support, this option provides messaging engine failover. For a high availability messaging WebSphere Application Server configuration, there is one messaging engine running at a time for each SIBus. The messaging engine can run in multiple application servers, specifically the other members of the messaging cluster. If the server in which it is currently running becomes unavailable, it is activated in another of the servers of the messaging cluster with which the messaging engine is associated. All messages are preserved because the messaging engine state is saved in Derby. Messaging engines are activated in different application servers. The advanced configuration scripts create both the appropriate schemas and the high availability group OneOfNPolicy core group policies for messaging engine election and activation.

**Note:** Multiple messaging engines can run in a given application server.

**Scalable messaging engine configuration**

Processed over standard Java Message Service (JMS) support, this option enables multiple messaging engines to run in a WebSphere Application Server SIBus at a time. Therefore, the message flows for the various JMS applications can be divided or partitioned across the different messaging engines. Scalable messaging provides a greater number of messages that are processed by the WebSphere Application Server JMS support and therefore a scalable implementation.
The advanced configuration scripts create multiple messaging engines for the SIBus, specifically one for each member of each HVMsgCluster. The default is one cluster and two members. Then, multiple messaging engines are activated when HVMsgCluster members are started. The OneOfNPolicy core group policies are created for each messaging engine ensuring the messaging engine to cluster member mapping.

The appropriate schemas and high availability group policies for messaging engine election and activation are also created.

**Note:** Multiple messaging engines can run in a given application server.

**Highly available and scalable messaging engine configuration**

Provides multiple messaging engines to run in multiple cluster members for each SIBus. The provided scripts handle the schema definitions, core group policies, and message engine to member mappings. With this configuration, messaging traffic can be divided across multiple messaging engines. If an application server goes down, any messaging engines are activated in the remaining cluster members.

**Note:** Multiple messaging engines can run in a given application server.

**Enable MQ messaging**

In addition to these messaging options, the **Enable MQ messaging** option provides some of the traditional WebSphere MQ configuration options available with WebSphere Application Server. It provides additional MQ link configuration on the deployment manager part.

**Note:** Newer features in WebSphere Application Server that use WebSphere MQ as an external Java message service (JMS) provider are available. These features are based on how you create your JMS application resources. However, if your messaging application does not support this approach, IBM Cloud Orchestrator provides features based on server configuration. For details, see the information about configuring JMS resources for WebSphere MQ messaging provider in the WebSphere Application Server information center.

The following options provide configuration for two servers:

**MQ link configuration**

Select the WebSphere MQ link configuration option to perform the following function:

- Create new transport chains for the WebSphere MQ link and associate them with each messaging engine. These transport chains are basic if no security exists or SSL if security is enabled.
- Create the WebSphere MQ link
- Create the foreign bus and associate it with the WebSphere MQ link

When defining the WebSphere MQ link there are items that use WebSphere MQ configuration attributes. You must adjust these attributes, for example the sample host, port, and user IDs, to reflect your actual WebSphere MQ environment.
MQ server configuration

Select the WebSphere MQ server configuration option to perform the following function:

- Create new transport chains for the WebSphere MQ server and associate them with each messaging engine. These transport chains are basic if no security exists or SSL if security is enabled.
- Create the WebSphere MQ server
- Add the WebSphere MQ server to the SIBuses

When defining the WebSphere MQ server there are items that use WebSphere MQ configuration attributes. You must adjust these attributes, for example the sample host, port, virtual queue manager name, and user IDs, to reflect your actual WebSphere MQ environment.

3. Enable session persistence. HVWebCluster or application clusters are created with associated replication domains. Therefore, you can use the hyper text transfer protocol (HTTP) session replication. If the replication domain is defined, no resources are created or used unless session replication is configured. You can use the Enable session persistence option and then one of the following options to use HTTP session persistence:

Memory-memory implemented session persistence

The HTTP session memory bit is set on all the HVWebCluster servers.

Database implemented session persistence

On the virtual system instance, the JDBC data source created must be updated on the deployment manager with valid host, port, user name, and password values. The appropriate client drivers for your database, for example jars and libraries, must be installed on your WebSphere systems.

To use this option on the virtual system instance, the JDBC data source created must be updated on the deployment manager. JDBC data source must have valid host, port, user name, and password values. The appropriate client drivers for your database, for example jars and libraries, must be installed on your WebSphere Application Server systems. IBM Cloud Orchestrator performs the following operations:

- Creates a DB2 JDBC provider
- Creates a sample DB2 data source, with dummy values for the host, port, ID, and password
- Sets up a session manager on each HVWebCluster server to do HTTP session to the database, using the sample data source and dummy connection values

For important information about supplying a database that an HTTP session over the database supports, see "Configuring database implemented session persistence for Derby" on page 439.

4. Enable global security. Use the global security option to perform the following function:

- Set the global security admin bit
- Use the WIM user registry that is provided with IBM Cloud Orchestrator
- Use both LTPA and BasicAuth for authentication (BasicAuth is needed for stand-alone clients)
- Use an SSL-allowed policy for CSIv2
• Turn off single sign-on interoperability
• Configure secure file transfer between the deployment manager and the node agents
• Use the high availability manager for the secure DCS channel

Using global security provides the option to use secure messaging. Use secure messaging to perform the following function:
• Set the security bit on the SIBus
• Reduce the set of users that can connect to the SIBus. Only the WebSphere Application Server ID created with the CB UI can connect. The default value for that ID is virtuser.
• Disable the InboundBasicMessaging transport for the messaging engines
• Adds the virtuser ID to the sender role for foreign buses for MQLink configurations

Results

You have configured the advanced options for the virtual system pattern.

What to do next

If you are editing parts for WebSphere advanced cluster or WebSphere advanced cluster (development) virtual system patterns from an WebSphere Application Server 7.0.0.17 with Intelligent Management Pack image, then there are more advanced options you can configure. For more information, see “Configuring advanced options for Intelligent Management Pack.”

Depending on the database you are using, you can configure advanced messaging. For more information, see “Configuring advanced messaging for databases” on page 436.

Configuring advanced options for Intelligent Management Pack:

If you are working with virtual system patterns from an IBM WebSphere Application Server Hypervisor Edition Intelligent Management Pack image, there are additional advanced options that you can configure.

Before you begin

When you open the advanced options editor for an Intelligent Management Pack virtual system pattern, default settings are shown for the virtual system pattern. These settings are recommended values for the topology. To accept these default values for this topology, click OK. To return to the virtual system pattern without setting these default values, click Cancel.

For information about basic advanced options configuration, see “Configuring advanced options” on page 425.

About this task

In addition to the basic advanced options described in “Configuring advanced options” on page 425, you can also configure advanced options specifically for Intelligent Management Pack virtual system patterns. Advanced options enable you to use a policy-based approach to managing your applications. You can enforce application health actions with no service disruption.
Procedure

1. **Define dynamic clusters.** Select this option to begin to define dynamic clusters across the custom node parts in the virtual system pattern.

   a. **Create dynamic clusters.** Select this option to create dynamic clusters across all custom node parts in the virtual system pattern. You can set values for the following parameters:
      - `DYNAMIC_CLUSTER_PREFIX`
      - `NUMBER_OF_DYNAMIC_CLUSTERS`
      - `MAXIMUM_INSTANCES_PER_NODE`
      - `MAXIMUM_NODES`
      - `MINIMUM_TOTAL_INSTANCES`
      - `MAXIMUM_TOTAL_INSTANCES`
      - `SERVER_INACTIVITY_TIME`: Specify the time unit in minutes.

   b. **Create ODR dynamic clusters.** Select this option to create on demand router (ODR) dynamic clusters across all ODR nodes in the virtual system pattern.

   c. **Enable elasticity mode.** Select this option to enable the elasticity mode of the application placement controller. You can set values for the following parameters:
      - `ELASTICITY_MODE`: When the reaction mode is set to automatic, no user input is required for the associated task, such as adding or removing instances, to be carried out. When the mode is set to supervised, a runtime task that proposes one or more reactions is created. The system administrator can approve or deny the task.
      - `ELASTICITY_OPERATIONS_TIMEOUT`: This value defines the allotted period of time during which a task is completed. The default value is 120 minutes. See “Configuring elasticity mode and the associated operations” on page 434 for configuration details.

2. **Enable overload protection.** Using overload protection, you can specify processor and memory overload protections. Processor and memory thresholds are enforced by the Autonomic Request Flow Manager (ARFM) controller component of Intelligent Management Pack. Processor and memory overload protections are functions of ARFM.

   a. **Memory overload protection:** Controls the rate at which requests without affinity are permitted through the ODR. Use memory overload protection to prevent Java heap utilization from exceeding a threshold that you can specify. This option adds the heap overload protection configuration script to the ODR. The heap overload protection configuration script can be configured by specifying the `PERCENTAGE_OF_MAXIMUM_HEAP_SIZE` parameter, which is the maximum rate (in calls per second) that can be sustained without exceeding the percentage of the maximum heap size.

   b. **CPU overload protection:** Controls the rate at which requests without affinity are permitted through the ODR. Setting the processor overload protection prevents processor utilization from exceeding a threshold that you can specify. This option adds the CPU overload protection configuration script to the ODR. Configure the configuration script by specifying the `MAXIMUM_CPU_USAGE` parameter.

3. **Configure standard health policies.** Health policies are the definitions of specific health criteria that you want your environment to protect itself against. Use this option to configure the following health policies:
a. **Excessive heap usage**: The excessive heap usage health policy is triggered when memory usage exceeds the specified percentage of the heap size for a specified time. Selecting this option adds the excessive memory usage policy configuration script to the deployment manager part. You can configure the excessive memory usage health policy by specifying the following script parameters:

- `HEAP_USAGE_PERCENTAGE`
- `OFFENDING_TIME_PERIOD`
- `OFFENDING_TIME_UNIT`: Specify the time unit in minutes.
- `EXCESSIVE_MEMORY_USAGE_POLICY_REACTION_MODE`
- `EXCESSIVE_MEMORY_USAGE_POLICY_ACTION`
- `EXCESSIVE_MEMORY_USAGE_POLICY_NAME`

b. **Memory leak**: Starts when a memory leak is detected. This policy checks if trends, in the free memory that is available to the server in the Java heap, decrease over time. This option adds the memory leak policy configuration script to the deployment manager. Configure the configuration script by specifying the following parameters:

- `MEMORY_LEAK_DETECTION`
- `MEMORY_LEAK_POLICY_REACTION_MODE`
- `MEMORY_LEAK_POLICY_ACTIONS`
- `MEMORY_LEAK_POLICY_NAME`

c. **Maximum server age**: Starts after an application server has been running for a specified amount of time. This option adds the maximum server age policy configuration script to the deployment manager. Configure the configuration script by specifying the following parameters:

- `SERVER_AGE`
- `SERVER_AGE_UNIT`
- `MAXIMUM_SERVER_AGE_POLICY_REACTION_MODE`
- `MAXIMUM_SERVER_AGE_POLICY_ACTIONS`
- `MAXIMUM_SERVER_AGE_POLICY_NAME`

d. **Email notification list**: Specifies a list of email addresses to receive notification when a health condition is met. This option adds the email notification configuration script to the deployment manager. Configure the configuration script by specifying the following parameters:

- `SMTP_HOST_NAME`
- `SMTP_PORT`
- `SMTP_USERID`
- `SMTP_PASSWORD`
- `EMAIL_ADDRESSES_TO_NOTIFY`
- `SENDER_ADDRESS`

4. **Configure ODR-dependent health policies**: Select this option when there is an ODR in the virtual system pattern. Use this option to configure the following health policies:

a. **Maximum requests served**: Starts after an application server has served a specified number of requests. Configure the configuration script by specifying the following parameters:

- `TOTAL_REQUESTS`
- `MAXIMUM_REQUESTS_POLICY_REACTION_MODE`
- `MAXIMUM_REQUESTS_POLICY_ACTIONS`
b. **Excessive number of timed out requests**: Starts after a specified number of requests timeout within a 1 minute interval. Configure the configuration script by specifying the following parameters:
   - `REQUEST_TIMEOUT_PERCENTAGE`
   - `EXCESSIVE_REQUEST_TIMEOUT_POLICY_REACTION_MODE`
   - `EXCESSIVE_REQUEST_TIMEOUT_POLICY_ACTIONS`
   - `EXCESSIVE_REQUEST_TIMEOUT_POLICY_NAME`

c. **Excessive average response time**: Starts when the average response time exceeds a specified response time threshold. Configure the configuration script by specifying the following parameters:
   - `EXCESSIVE_RESPONSE_TIME`
   - `EXCESSIVE_RESPONSE_TIME_UNIT`
   - `EXCESSIVE_RESPONSE.TIME_POLICY_REACTION_MODE`
   - `EXCESSIVE_RESPONSE_TIME_POLICY_ACTIONS`
   - `EXCESSIVE_RESPONSE_TIME_POLICY_NAME`

d. **Storm drain detection**: This policy tracks requests that have a decreased response time which has been predetermined as a significant decrease. The actions specified for this policy are run and the associated server is restarted when the specified detection level is reached. Configure the configuration script by specifying the following parameters:
   - `STORM_DRAIN_DETECTION_LEVEL`
   - `STORM_DRAIN_POLICY_REACTION_MODE`
   - `STORM_DRAIN_POLICY_ACTIONS`
   - `STORM_DRAIN_POLICY_NAME`

**Results**

You have configured the Intelligent Management Pack advanced options for the virtual system pattern.

**What to do next**

Depending on the database you are using, you can configure advanced messaging. For more information, see "Configuring advanced messaging for databases" on page 436.

**Configuring elasticity mode and the associated operations**:

Configure elasticity mode to add logic that causes the application placement controller to minimize the number of nodes that are used, as well as remove nodes that are not needed, while still meeting service policy goals. Additionally, you can configure elasticity mode to add logic so that when the controller recognizes a particular dynamic cluster is not meeting service policies and has started all possible servers, the controller calls to add a node.

**Before you begin**

- Select the **Enable elasticity mode** option in the advanced options editor as described in "Configuring advanced options for Intelligent Management Pack" on page 431.
- For optimal performance, ensure that your dynamic clusters are running in supervised mode or automatic mode. It is not recommended to have elasticity
mode enabled when your dynamic clusters are running in manual mode. However, if the dynamic clusters are running in manual mode with elasticity enabled, consider the following items:

- The application placement controller does not add nodes to dynamic clusters in manual mode.
- The application placement controller does not remove nodes from dynamic clusters in manual mode when a server is started on the specific nodes.
- The application placement controller does remove nodes from dynamic clusters in manual mode when a server is not started on the specific nodes.

- It is not recommended to use elasticity mode with uncapped mode.
- It is not recommended to enable elasticity mode when the following option is set in the administrative console for one or more dynamic clusters: If other dynamic clusters need resources, stop all instances of this cluster during periods of inactivity. If you have elasticity mode enabled and the option set, the application placement controller can remove all of the custom nodes in the cell.
- Configure certain controllers to start on the deployment manager or node agent that will not be removed.

1. To configure the application placement controller to start on the deployment manager, select **System administration > Deployment manager > Java and process management > Process definition > Java virtual machine > Custom properties**.
   a. Enter the name of the custom property as `HAManagedItemPreferred_apc`.
   b. Set the value of the custom property to true.
   c. Click **Apply**, and save your changes.
   d. Restart the current process in which the application placement controller is running.

2. To configure the application placement controller to start on one of the nodes that contains an ODR, select **System administration > Nodes > node_name > node_agent_name > Java and process management > Process definition > Java virtual machine > Custom properties**.
   a. Enter the name of the custom property as `HAManagedItemPreferred_apc`.
   b. Set the value of the custom property to true.
   c. Click **Apply**, and save your changes.
   d. Restart the current process in which the application placement controller is running.

3. When you use elasticity mode in an environment in which multi-cell performance management is configured, you must configure certain controllers to start on the deployment managers of the center cell and the point cells.
   a. Set the `HAManagedItemPreferred_apc` custom property to true on the deployment manager of the center cell.
   b. Set the `HAManagedItemPreferred_cellagent` custom property to true on the deployment manager of the point cells.

**About this task**

When you enable elasticity mode in the advanced options editor, the following default actions are associated with the add and remove operations. The elasticity operations define the runtime behaviors to monitor, and the corrective actions to take when the behaviors are present.

1. **Add virtual machine**: Creates and federates a new node into the cell
2. **Add node to dynamic cluster action**: Adds a newly-created node to the dynamic cluster membership.

3. **Remove node from cell**: Removes the node from the cell.

4. **Remove virtual machine**: Removes the virtual machine from the associated hypervisor.

Complete the following procedure to define additional custom actions for the add and remove operations.

**Procedure**

1. Select **Operational policies > Autonomic controllers > Application placement controller > Elasticity operation**. Select the operation.

2. To add additional actions to the add operation, click **Add Action...** Select the custom action from the list of Custom elasticity operation actions. If no custom actions are defined, select **Operational policies > Autonomic controllers > Application placement controller > Elasticity Custom Actions > New**.

3. To add additional actions to the Remove operation, click **Add Action...** Select the custom action from the list of Custom elasticity operation actions. If no custom actions are defined, select **Operational policies > Autonomic controllers > Application placement controller > Elasticity Custom Actions > New**.

**Configuring advanced messaging for databases:**

When you configure the advanced options on a cluster virtual system pattern and you want to enable messaging, there are some additional configuration options. You can use these options to start and use clustered messaging engines or to use a sample authentication alias for databases.

**Before you begin**

See the information about working with cluster virtual system patterns that is provided in ["Configuring advanced options for cluster virtual system patterns (classic)" on page 427](#).

**About this task**

To work with clustered messaging engines or configure a sample authentication alias for databases, in certain WebSphere Application Server configurations, further configuration is required.

**Procedure**

- Use the IBM Cloud Orchestrator sample authentication alias. The sample authentication alias can be used by the messaging engines for databases other than Derby. If you use the IBM Cloud Orchestrator messaging recommendations and a database, for example DB2 or Oracle, perform the following steps:
  1. Create the appropriate Java Database Connectivity (JDBC) driver and data source for your database.
  2. Update the sample authentication alias for the database provided by IBM Cloud Orchestrator to contain the correct credentials to your database.
  3. Change the messaging engine and SIBus configuration to use the new data source, where the updated authentication alias is used with this data source.
- Use a database server. To correctly start and use clustered messaging engines, a database server is required. You can use the Derby database server shipped with
WebSphere Application Server. IBM Cloud Orchestrator provides configuration, the definition of a Derby database server shipped with WebSphere Application Server, which runs on the deployment manager of the virtual system pattern. A JDBC provider and data source point to a Derby network server. This server is started from the deployment manager node, from the Derby installation under the WAS installation root. IBM Cloud Orchestrator provides this function for messaging to minimize later configuration. To start Derby from the deployment manager before starting the messaging cluster, update the Derby configuration for remote connections. To update this process, perform the following steps:

1. Edit the \(<WAS\_HOME>\)/derby/derby.properties file. Uncomment the following line:
   ```
   #derby.drda.host=0.0.0.0
   ```
2. Start the Derby network server. To start this server, from the
   ```
   \(<WAS\_HOME>\)/derby/bin/networkServer/ directory run the following command:
   ```
   ```startNetworkServer.sh | .bat```
3. Start the Derby database server on the deployment manager node.

**Results**

You have configured the cluster virtual system pattern to start and use clustered messaging engines or to use a sample authentication alias for databases.

**What to do next**

When you have configured messaging, you can configure the advanced options for the cluster virtual system pattern you are editing.

**Configuring advanced options for single server virtual system patterns (classic):**

You can use the advanced virtual system pattern configuration function as a starting point in configuring the single server virtual system pattern that defines your cell.

**Before you begin**

When you open the advanced options editor for a new single server virtual system pattern, default settings are shown for the virtual system pattern. These settings are recommended values for the topology. To accept these default values for this topology, click OK. To return to the virtual system pattern without setting these default values, click Cancel.

**About this task**

The following options are available to further define single server virtual system patterns:

- Enable session persistence
- Global security

**Procedure**

1. Enable the session persistence. HVWebCluster or application clusters are created with associated replication domains. Therefore, you can use the hyper text transfer protocol (HTTP) session replication. If the replication domain is defined, no resources are started or used unless session replication is
configured. You can use the **Enable session persistence** option and then one of the following options to configure HTTP session persistence:

**Memory-memory implemented session persistence**
The HTTP session memory bit is set on all the HVWebCluster servers.

**Database implemented session persistence**
To use this option on the virtual system instance, the Java Database Connectivity (JDBC) data source created must be updated on the deployment manager. You must provide valid host, port, user name, and password values. Also, the appropriate client drivers for your database, for example jars and libraries, must be installed on your WebSphere Application Server systems. IBM Cloud Orchestrator performs the following operations:

- Creates a DB2 JDBC provider
- Creates a sample DB2 data source, with dummy values for the host, port, ID, and password
- Sets up a session manager on each HVWebCluster server to do HTTP session to the database, using the sample data source and dummy connection values

For important information about supplying a database that an HTTP session over the database supports, see "Configuring database implemented session persistence for Derby" on page 439.

2. Enable global security. Using global security provides the following functions:

- Sets the global security admin bit
- Uses the WIM user registry that is provided with IBM Cloud Orchestrator
- Uses both LTPA and BasicAuth for authentication (BasicAuth is needed for stand-alone clients)
- Uses an SSL-allowed policy for CSIv2
- Turns off single sign-on interoperability
- Configures secure file transfer between the deployment manager and the node agents
- Enables the high availability manager to use the secure DCS channel

Use the global security option to configure secure messaging. Secure messaging provides the following function:

- Sets the security bit on the SIBus
- Reduces the set of users that can connect to the SIBus. Only the WebSphere Application Server ID created with the CB UI can connect. The default value for that ID is `virtuser`.
- Disables the InboundBasicMessaging transport for the messaging engines
- Adds the `virtuser` ID to the sender role for foreign buses for MQLink configurations

**Results**

You have configured the advanced options for a single server virtual system pattern.

**What to do next**

You can run the `wasCBUpdateSessDSInfo.py` script, to configure database implemented session persistence. For more information, see "Configuring database implemented session persistence for Derby" on page 439.
implemented session persistence for Derby.”

Configuring database implemented session persistence for Derby:

Because IBM Cloud Orchestrator does not supply a database that an HTTP session over the database supports, you must supply connection information for your database.

About this task

An WebSphere Application Server HTTP Session over the database does not support the use of Derby. Therefore you must supply connection information to configure database implemented session persistence for either a cluster or single server virtual system pattern. If you are using DB2, then you can update the provided data source and session manager for each HVWebCluster server using the `wasCBUpdateSessDSInfo.py` script.

Procedure

1. Use the `wasCBUpdateSessDSInfo.py` script. To configure database implemented session persistence, run the `wasCBUpdateSessDSInfo.py` script using the following parameters:
   - `-dbHost <host name>`
   - `-dbPort <port number>`
   - `-dbUser <user ID>`
   - `-dbPassword <password of the database user>`

2. Ensure that this script is in the correct directory. After the virtual system pattern is deployed, ensure that this script is in the `<WAS profile root>/bin/DeployerScripts` folder.

Results

When you have run the `wasCBUpdateSessDSInfo.py` script, database implemented session persistence can be enabled.

Making virtual system patterns (classic) read-only

Either draft or read-only virtual system patterns can be deployed for testing or production, but making a virtual system pattern read-only prevents further edits to the topology definition. Making virtual system patterns read-only provides consistent reuse in the cloud.

Before you begin

You must have edit access to any virtual system pattern you want to make read-only. If it is read-only, it is already locked to editing and cannot be changed. To make a virtual system pattern read-only, first be sure that no one with permission to edit the virtual system pattern intends to change it.

About this task

You can make virtual system patterns read-only if you want to prevent further editing to the virtual system pattern.
Procedure

1. From the left panel of the Virtual System Patterns (Classic) window, select the virtual system pattern. Virtual system patterns that have the read-only symbol by them are already read-only and cannot be edited. Virtual system patterns with the edit symbol beside them are not read-only and can be edited. Basic information about the selected virtual system pattern is shown in the right panel of the Virtual System Patterns (Classic) window.

2. Determine if you are ready to lock editing of the virtual system pattern.
   - If virtual system pattern editing is complete and you are ready to make the virtual system pattern read-only, click the Lock icon in the upper right toolbar of the Virtual System Patterns (Classic) window.
   - If virtual system pattern editing is not complete, see the information in “Editing a virtual system pattern (classic)” on page 418 and “Configuring advanced options” on page 425.

3. Verify that you want to make the virtual system pattern read-only. When prompted to verify that you want to make the virtual system pattern read only, click OK to lock the virtual system pattern.

Results

When you have made the virtual system pattern read-only, it can be cloned or deleted but it cannot be edited.

What to do next

You can deploy the virtual system pattern to the cloud. For more information, see “Deploying a virtual system pattern (classic).”

Deploying a virtual system pattern (classic)

You can deploy virtual system patterns to run in a cloud group. You can deploy either draft or committed virtual system patterns for testing or production.

Before you begin

Configure the virtual system pattern, including the advanced configuration, and ensure that it is ready to be deployed. See “Editing a virtual system pattern (classic)” on page 418 and “Configuring advanced options” on page 425 for more information.

When a virtual system provisioning is requested by a user in a project and targeted to an environment profile that maps to an OpenStack region, the quota limitations are checked in the following sequence:

1. Workload Deploer checks that the requested capacity fits the environment profile quota.
2. OpenStack checks that the required capacity fits the user’s project quota.

If both checks are passed, then the virtual system deployment is performed. To modify quotas on OpenStack, refer to “Configuring project quotas” on page 225. To modify quotas in the environment profile, refer to “Creating an environment profile” on page 345.

About this task

This task describes deploying a virtual system pattern to run in the cloud group.
Procedure

1. From the list in the left panel of the Virtual System Patterns window, select the virtual system pattern to deploy.

2. Indicate that you want to deploy the virtual system pattern. Click the **Deploy** icon on the upper right panel of the Virtual System Patterns (Classic) window.

3. Provide the necessary information. The Describe the virtual system instance you want to deploy dialog provides the fields of information to deploy the virtual system pattern to the cloud. The parameters that are required differ depending on any advanced configuration you have defined and any associated script packages you have included. Links to the advanced configuration and the scripts are provided on the interface. Provide the following information to deploy the virtual system pattern:

   **Virtual system instance name**
   Enter the name of the virtual system instance in which to deploy this virtual system pattern.

   **Choose Environment**
   You can deploy the virtual system pattern using an environment profile. Make your selections from the following options:

   - **IP version**
     IPv4 is selected. IPv6 is not currently supported.

   - **Choose cloud group**
     This option is not currently supported.

   - **Choose profile**
     Select this option to deploy the virtual system pattern using an environment profile. Then select the environment **Type** and a valid environment **Profile** from the lists.

     **Note:** If the **Pattern deployer** option was chosen, you cannot specify an IP address that is contained within the IP groups which are defined in IBM Cloud Orchestrator.

     **Note:** IBM Cloud Orchestrator is not able to filter environment profiles suitable for deployment to VMware clusters based on the images contained in the virtual system pattern that you are deploying. Make sure you are selecting a valid environment profile.

     For more information about environment profiles, see [Managing environment profiles](#) on page 344

   **Schedule deployment**
   Click this link to provide information about when the virtual system pattern is to be deployed and for how long. You can deploy the virtual system pattern immediately after providing the information in the dialog or you can schedule deployment using the following options:

   - **Start now**
     Deploys the virtual system pattern immediately after providing the required information in the dialog. **Start now** is the default option.

   - **Start later...**
     Provide the date and time to deploy this virtual system pattern at a later time.
Run indefinitely
   Runs this virtual system pattern continuously. **Run indefinitely**
   is the default option.

Run until...
   Use this option to provide the end date and time for the virtual
   system pattern to stop running.

Configure parts
   For each virtual image part requiring information, click the link and
   provide the information for each configuration parameter shown. The
   set of parameters depends on the part itself. The parts that require
   information are different depending on the type of virtual image to be
   deployed and the type of hypervisors in the cloud. For example, parts
   for an WebSphere Application Server image would require different
   information than parts for a DB2 image.

**Note:** The administrator password that you specify for a Windows
virtual image must meet complexity requirements (for example,
Password).

**Note:** The WebSphere administrative user name should be non-root
user only.

**Note:** If you are using a non-English operating system, you must
specify the correct language and country in the **Default Locale**
and **Default Country** parameters to correctly deploy the virtual image.

For virtual image parts, you must specify a value in the **Flavor** field. In
OpenStack, the instance flavor describes the memory and storage
capacity of the virtual machine to be deployed. By default the flavor
values are:

**m1.tiny**
   Memory: 512 MB, vCPUs: 1, Storage: 0 GB

**m1.small**
   Memory: 2048 MB, vCPUs: 1, Storage: 20 GB

**m1.medium**
   Memory: 4096 MB, vCPUs: 2, Storage: 40 GB

**m1.large**
   Memory: 8192 MB, vCPUs: 4, Storage: 80 GB

**m1.xlarge**
   Memory: 16384 MB, vCPUs: 8, Storage: 160 GB

**Note:**
   • The flavor values might change depending on the configuration of
     your OpenStack environment. In OpenStack, use the nova
     flavor-list command to view the list of available flavors and their
     characteristics.
   • The 0 GB storage size is a special case which uses the native base
     image size as the size of the ephemeral root volume. When you use a
     flavor with 0 GB of storage, no automatic check is performed by
     OpenStack on available storage capacity. You must ensure that there
     is sufficient storage to contain the provisioned VM image disks.
To ensure that proper storage capacity check is performed by OpenStack, use flavors that specify storage size greater than zero and larger than the image disk size. Note that on VMware the image uses IDE driver. Then, VMware cannot perform disk expansion and the flavor storage size must match the disk size of the provisioned image or specify disk=0.

- To understand what memory, CPU and disk size should be used while creating a new flavor, refer to “Creating new flavors in OpenStack” on page 106.
- Only flavors that are suitable for the selected images are displayed.

For information about changing the virtual machine flavor after deploying the virtual system pattern, see “Virtual machine panel fields on the Virtual System Instances (Classic) window” on page 464.

If the information for each of these virtual image parts is provided, a green check mark is shown to the left of the virtual image part. If there is information missing for one of these virtual image parts, the check box to the left of it does not contain a green check mark. In this case, click the virtual image part missing information and provide that information in the fields shown.

If you are using an environment profile there might be additional fields to configure for the parts. If the environment profile specifies that the virtual system pattern deployer is to provide the IP address, the IP addresses must also be provided for the parts. Specify the following part information:

**In cloud group**
Select the cloud group as you normally would. If an alias was provided to define the cloud group in the environment profile, then the alias name is available to be selected in this field.

**IP Group**
Select an IP group as you normally would. If an alias was provided to define the IP group in the environment profile, then the alias name is available to be selected in this field.

**Addresses:**
Provide both the host name and the IP address. The host name and IP address must not exist in the selected IP group.

If you are deploying parts with add-ons, you can configure fields for those add-ons if they were not configured when the part was created and locked to editing during deployment.

4. Deploy the virtual system pattern. When all of the information is provided correctly in the dialog, click OK to deploy the virtual system pattern.

**Results**

The virtual system pattern is deployed to the cloud and runs in the selected virtual system instance.

On a topology deployment, some parts can reserve CPU or memory, or both. These fields effect how the CPU and memory are configured on the underlying hypervisor. The CPU and memory limits are set and reserved for ESX hypervisors. This setting prevents the CPU from being overcommitted but reduces license usage.
What to do next

To add additional nodes to the virtual system pattern, first stop the virtual system in which the cloud the virtual system pattern is deployed to is running. For more information about virtual system instances, see “Managing virtual system instances (classic)” on page 452.

Deploying a pattern (classic) with additional actions:

You can deploy a virtual system pattern with additional configuration options that were previously defined in Business Process Manager.

Procedure
1. From the list in the left panel of the Virtual System Patterns (Classic) window, select the virtual system pattern to deploy.
2. Click the Deploy in the cloud icon on the upper right panel of the Virtual System Patterns (Classic) window. A popup window opens with a set of deployment options and parameters that you can configure.
3. Select options for the virtual system that you want to deploy. For information about specific settings, see “Deploying a virtual system pattern (classic)” on page 440.
4. If actions with user interface are defined on this pattern, a Configure actions section is available. Click the link to view all actions that must be configured.
5. Click the action that you want to configure, and submit any required parameters. A green checkmark is displayed next to the action name.
6. Repeat the previous step for all other actions that require configuring.
7. When all of the information is provided correctly in the dialog, click OK to deploy the virtual system pattern.

Deleting a virtual system pattern (classic)

You can delete any virtual system patterns you own using the IBM Cloud Orchestrator user interface.

Before you begin

You must be the owner of a virtual system pattern you want to delete.

About this task

You can delete a custom virtual system pattern if the access control lists (ACLs) permit you to do so.

Procedure
1. From the list of virtual system patterns in the left panel of the Virtual System Patterns (Classic) window, select the virtual system pattern to delete. If the virtual system pattern is not shown, you can also search for a virtual system pattern using the search function. Basic information about the selected virtual system pattern is shown in the right panel of the Virtual System Patterns (Classic) window.
2. Determine if you are ready to delete the virtual system pattern.
   - If you are ready to delete the virtual system pattern, click the delete icon in the upper right toolbar of the Virtual System Patterns window.
If you want to change the virtual system pattern instead of deleting it, see
the information in “Editing a virtual system pattern (classic)” on page 418
and “Configuring advanced options” on page 425.

3. Verify that you want to delete the virtual system pattern. When the prompted
to verify that you want to delete the virtual system pattern, click OK.

Results

You return to the Virtual System Patterns (Classic) window and the virtual system
pattern has been deleted and is no longer shown in the list on the left panel.

Virtual system pattern (classic) windows

The Virtual System Patterns (Classic) and Pattern Editor windows provide fields to
view and work with your virtual system pattern topology. Virtual system patterns
contain parts, scripts, and add-ons that you can graphically add and edit to
customize your topology.

There are two interactive windows for working with virtual system patterns:

• The Virtual System Patterns (Classic) window provides the following interactive
  panels:

  Left panel
  The left panel of the Virtual System Patterns (Classic) window lists the
  virtual system patterns available for the virtual images. You can also use
  this panel to search for or add new virtual system patterns.

  Right panel
  The right panel of the Virtual System Patterns (Classic) window shows
  information, including the topology, about a virtual system pattern you
  select from the list on the left.

  For information about the fields on the Virtual System Patterns (Classic)
  window, see “Fields on the Virtual System Patterns (Classic) window” on page
  446.

• The Pattern Editor window provides the following interactive panels:

  Left panel
  When a specific virtual system pattern is being edited, the Pattern Editor
  panel provides lists of scripts, parts, and add-ons that can be added to
  the virtual system pattern topology.

  Right panel
  Virtual system pattern parts, scripts, and add-ons associated with the
  parts can be edited graphically on this canvas.

  For more information about the fields on the Pattern Editor window, see “Fields
  on the Pattern Editor window” on page 448.
Fields on the Virtual System Patterns (Classic) window:

The IBM Cloud Orchestrator Virtual System Patterns (Classic) window lists the virtual system patterns you have configured and provides fields to view and work with your topology. The Virtual System Patterns (Classic) window graphically shows the parts, scripts, and add-ons in your topology.

There are two interactive panels of the Virtual System Patterns (Classic) window. The virtual system patterns are listed in the left panel. Configuration and topology information for a selected virtual system pattern is displayed in the right panel.

Virtual system patterns list

The left panel of the Virtual System Patterns (Classic) window provides a list of virtual system patterns to work with. Virtual system patterns that have the read-only symbol beside them are read-only and cannot be edited. To work with one of these virtual system patterns, it must be cloned and then the clone can be edited. Virtual system patterns in the list that have the draft symbol beside them can be edited directly.

An Add icon at the top of the window adds a new virtual system pattern. See “Creating a virtual system pattern (classic)” on page 416 for more information. A search and sort function searches for virtual system patterns not listed and sorts through the list of virtual system patterns that are found.

Icons on the Virtual System Patterns (Classic) window

The following icons are on the upper right top bar:

Refresh
Refreshes the virtual system pattern display in the configuration panel after any changes, such as editing or deployment.

Deploy
Deploys the virtual system pattern to the cloud group you specify. This icon is available when the virtual system pattern contains at least one virtual part. Click it to specify the cloud group and deploy the virtual system pattern. See “Deploying a virtual system pattern (classic)” on page 440 for more information about deploying virtual system patterns.

Edit
This icon is available if the virtual system pattern is not read-only and can be edited. See “Editing a virtual system pattern (classic)” on page 418 for more information.

Clone
Clones the selected virtual system pattern. Cloning a virtual system pattern is useful if the virtual system pattern is read-only and cannot be edited. This option is available for the predefined virtual system patterns that are provided by IBM and any virtual system patterns that are created and locked to editing. You can clone the virtual system pattern and then edit and deploy the copy. See “Cloning an existing virtual system pattern (classic)” on page 414 for more information about cloning virtual system patterns.

Lock
Locks the virtual system pattern to editing, making it read-only. See “Making virtual system patterns (classic) read-only” on page 439 for more information.

Delete
Deletes the virtual system pattern from IBM Cloud Orchestrator. See
Fields on the right configuration panel

Selecting a virtual system pattern in the left panel of the Virtual System Patterns (Classic) window displays the basic virtual system pattern properties in the right panel. The name of the selected virtual system pattern is displayed on the top bar of the right panel.

The following fields are available to specify and view some details about a virtual system pattern:

**Description**
An editable field that provides the description of the virtual system pattern. This description can help identify the virtual system pattern to other users who have been provided access to it.

**Created on**
Provides the time stamp when the virtual system pattern was created.

**Current status**
Provides the status of the virtual system pattern:
- **Draft** For any virtual system pattern you are creating, the initial status is Draft.
- **Locked (read only)** This field shows if the virtual system pattern is read only. Virtual system patterns can be made read only by clicking the lock icon on the top of the right panel.

**Updated on**
The timestamp of the most recent update.

**In the cloud now**
When the virtual system pattern is in use, this field shows the names of the virtual system instances currently running that were created from this virtual system pattern. Until you run the virtual system pattern in a cloud, this field displays (none) initially.

**Access granted to**
This field can be edited and it provides access to this virtual system pattern for other projects. Selecting projects, makes the virtual system pattern readable or writable to the users belonging to these projects. Initially this field is set to the role of the owner of the virtual system pattern.

By default, the Add more box contains the Everyone built-in project. When a project has been added, click the link beside the entry to toggle between the following access levels:
- Read
- Write
- All

Click the link name of the project to show information about that project. You can also click the remove link to remove access for a project.
Topology for this pattern
Displays any warnings or errors. This section of the panel provides a graphical representation of the topology.

Hypervisor type
Displays a message providing the type of hypervisor to which the virtual system pattern deploys.

Graphical topology
Provides a graphical display of the parts that make up the virtual system pattern. For WebSphere Application Server virtual system patterns, the parts in the topology can vary, depending on the virtual images you have installed.

To see the virtual image and operating system information for parts on the editing palette, hover your cursor over the part to display a pop-up window that provides this information.

Comments
Displays any comments that have been added to the virtual system pattern and provides a free form space to add comments.

For information about the Pattern Editor window, see “Fields on the Pattern Editor window.”

Fields on the Pattern Editor window:
The IBM Cloud Orchestrator Pattern Editor window contains lists of parts, scripts, and add-ons to graphically work with your topology. You can use these parts, scripts, and add-ons to edit and customize your virtual system pattern topology and, therefore, your deployment.

When a virtual system pattern is edited, using the edit icon in the Virtual System Patterns (Classic) window, the Pattern Editor window opens for that virtual system pattern. There are two interactive panels of the Pattern Editor window. The parts, scripts, and add-ons are listed in the left panel. The right panel is a virtual system pattern editor canvas for that specific virtual system pattern. The search function is available for parts, scripts, or add-ons.

Parts, scripts, and add-ons panel
The Parts, Scripts and Add-Ons lists are available in the left panel of the Pattern Editor window.

Parts
The Parts list displays the parts available to use in your virtual system pattern. The parts that are available depend on the virtual images you have installed and the hardware type of any parts already in the virtual system pattern. Only parts with the same hardware type as any parts already in the virtual system pattern are available. When you select parts in the Parts list, parts are then displayed. You can select them and drop them onto the canvas on the right side of the Pattern Editor.

Scripts
The Scripts list provides the script packages that are available. This list can contain any script packages that you have provided for use with IBM Cloud Orchestrator. You can add script packages to the parts on the editing palette. Add parts by dragging them onto the workspace on the right canvas of the Pattern Editor window and dropping them onto the part objects.
For more information about script packages, see "Adding a script package" on page 357 and "Associating a script package with a pattern" on page 361.

Add-Ons

The Add-Ons list provides the add-ons that are available. This list can contain any add-ons that you have provided for use with IBM Cloud Orchestrator. You can add add-ons to the nodes on the editing palette. Add the add-ons by dragging them onto the workspace on the right canvas of the Pattern Editor window and dropping them onto the node objects. The following types of add-ons can be added to the nodes:

- **Disk**: Add a disk to a node for deployment on a virtual machine.
- **NIC**: Add a NIC, or multiple NICs, to a node for deployment on a virtual machine.
- **User**: Add a user, or multiple users, to a node for deployment on a virtual machine.

For more information about add-ons, see "Adding add-ons to the catalog" on page 388.

See “Virtual system pattern (classic) editing views and parts” on page 419 for more information about the interaction of these parts on the canvas.

Available views

When a specific virtual system pattern is being edited in the Editor window, the graphical topology view is displayed in an editing canvas. There are two options to view a virtual system pattern on the editing canvas that are provided by toggle links at the top right of the page:

**Ordering**

This link changes the view to show the order the parts are started when the virtual system pattern is deployed. If you are working with a copy of a provided virtual system pattern, there is a recommended order and this order is the default setting. In this view, the parts are shown in the right column of the panel and numbered in the order they are started. The left column provides a textual description of the order in which the parts are started with order constraints for parts and scripts.

**Topology**

This link is shown when you are in the Ordering view. Click it to switch back to the topology view in which the relationship of the parts is shown.

Icons and links

From either the Topology or Ordering view, the following icons and links are on the upper right of the panel:

**Refresh**

Forces a refresh of the virtual system pattern to ensure that the diagram shows the current state of the virtual system pattern in IBM Cloud Orchestrator. Refreshing the virtual system pattern is useful if, for example, the virtual system pattern has been edited by another user since it was last retrieved by the web browser.
**Undo**  Undo the previous action. The virtual system pattern is saved during the editing process, so use this option to back up to the state of the virtual system pattern before the last edit.

**Undo all**
Undo all changes made in the current editing session for this virtual system pattern. The virtual system pattern is saved during the editing process, so use this option to back up to the state of the virtual system pattern before all edits were made during the current editing session.

**Done editing**
Indicates that you have finished editing and returns to the initial view of the virtual system pattern.

**Advanced options**
This link opens a configuration window for the virtual system pattern you are editing. The configuration window contains options that are available for this virtual system pattern. A set of default options are selected. These are common choices for topology virtual system patterns like the one you are editing. For more information about advanced options, see “Configuring advanced options” on page 425.

**Fields on the topology configuration panel**

The graphical editing canvas, on the right panel of the Pattern Editor window provides an interactive graphical display of parts. These parts, scripts, and add-ons make up the topology of the virtual system pattern. Parts displayed on the left panel can be added to the canvas and the parts on the canvas can be manipulated. The parts available can vary, depending on the virtual images you have installed. Available parts might include the following objects:

- Administrative agent
- Custom node
- Deployment manager
- IBM HTTP Server
- Job manager
- Stand-alone server
- On-demand router

Hovering your cursor over the part label displays a window that provides additional information about the part and its virtual image. The following actions can be performed:

- Drop parts onto the palette from the **Parts** list
- Edit the properties for the parts on the palette (using the edit properties icon on the part, to open a properties panel)
- Drop scripts or add-ons onto the parts from the **Scripts** and **Add-Ons** lists
- Edit parameters, if the script has parameters, using the edit properties icon.
- Change the count for some types of parts
- Lock the count
- Delete a part
- Change a part so that it comes from a different virtual image
- Delete scripts or add-ons
Virtual system pattern (classic) processing

When deploying a virtual system instance, that is a collection of virtual machines, IBM Cloud Orchestrator virtual system pattern processing follows a specific startup sequence.

A virtual system pattern is one or more virtual images and applications from the IBM Cloud Orchestrator catalog that implements a deployment topology.

When deploying a virtual system instance, a collection of virtual machines, it is helpful to understand the startup sequence that IBM Cloud Orchestrator virtual system pattern processing follows. The following sequence shows this order:

1. Virtual machines (defined by the virtual image parts) are started, to the extent possible, concurrently. Virtual image parts might have a dependency on other parts when cross configuration takes place as part of the virtual machine startup. See “Virtual system pattern (classic) editing views and parts” on page 419 for details about dependencies between virtual image parts. In such cases, the dependent virtual machines are started second. For example, in a WebSphere Application Server virtual system pattern, a deployment manager is started before any custom nodes. The custom nodes can be started either sequentially or concurrently with each other, depending on the image. WebSphere Application Server 6.1.0.25 custom nodes are started sequentially, for example, and WebSphere Application Server 7.0.0.5 nodes are started concurrently.

2. When the virtual machines are activated, custom nodes are federated by the deployment manager in the topology. If both stand-alone nodes and a deployment manager are in the topology, the stand-alone nodes are federated by the deployment manager. In addition, administrative agents and deployment managers are registered with a job manager if one is present in the topology.

3. Add-ons defined for each of the virtual machines are run.

4. The advanced setting scripts, provided by IBM that define the system configuration, run. This set of supplied scripts define settings like HTTP session clustering properties and security.

5. Script packages for each of the virtual machines are run. If a virtual system pattern specifies multiple script packages for the same virtual machine, the script packages are run in the order they were added to the virtual system pattern.

As an example, if you are deploying a virtual system pattern with a deployment manager and two custom nodes, the following processing occurs:

1. The deployment manager virtual machine and the deployment manager profile are started.

2. The first custom node virtual machine is started and the custom node profile is federated.

3. The second custom node virtual machine is started and the custom node profile is federated.
Managing virtual system instances (classic)

A virtual system instances is the virtual environment that is being managed by your IBM Cloud Orchestrator.

Before you begin

You must specifically be granted access to the virtual system instances or be assigned the admin role to perform these steps.

About this task

Virtual system instances are created when you deploy virtual images or use patterns composed of parts that are provided in your virtual images. The virtual images and the patterns are deployed to your hypervisors based on a component of IBM Cloud Orchestrator called placement. The internal placement component performs the job of deciding which hypervisors to use when deploying virtual machines. The placement component is also used when an existing virtual system instance is extended by adding virtual machines. It uses a complicated algorithm that considers a number of properties of the environment. For example, it considers the properties of the physical machines, existing virtual system instances on the hypervisors, and virtual machines on the hypervisor not managed by IBM Cloud Orchestrator. The properties of the virtual system instances being deployed or extended are also considered when making placement decisions. Most notably, the placement component considers the memory, physical CPUs, network addresses, disk space, and disk image sharing on the hypervisor. The placement component is part of the product code and is not configurable.

The virtual system instances are managed by IBM Cloud Orchestrator and can be serviced and accessed through the user interface. Virtual system instances managed by IBM Cloud Orchestrator are dynamic. Virtual machines can be added or removed to allow your virtual systems to scale based on current demand. If needed, you can scale down the environment and remove unnecessary virtual machines.

Procedure

You can use the user interface to manage your virtual system instance. See “Managing virtual system instances (classic) with the user interface” for more information.

Results

You have become familiar with all the actions associated with managing a virtual system instance.

Managing virtual system instances (classic) with the user interface

You can manage your IBM Cloud Orchestrator virtual system instances with the user interface.

Before you begin

You must specifically be granted access to the virtual system instance you intend to start or be assigned the admin role to manage your virtual system instances. To create a virtual system instance, you must deploy a pattern into the cloud. See “Deploying a virtual system pattern (classic)” on page 440 for more information.
about creating a virtual system instance by deploying a pattern.

**About this task**

You can perform a variety of tasks to manage your IBM Cloud Orchestrator virtual system instances with the user interface.

**Note:** Creating, restoring, or deleting snapshots for virtual system instances is not available on Power instances.

**Procedure**

1. Navigate to the Virtual System Instances (Classic) window by clicking PATTERNS > Instances > Virtual System Instances (Classic) from the menu bar.

   The list of the virtual system instances being managed by IBM Cloud Orchestrator is displayed along with the status of each virtual system instance. This list provides an overview of the existing virtual system instances but most management functions for these virtual system instances require you to select a specific virtual system instance.

2. Select a specific virtual system instance to manage by clicking a `<virtual_system_name>` from the list of the virtual system instances. The details for the virtual system instance you selected are displayed. If you want to manage a different virtual system instance, then click a different `<virtual_system_name>` and the associated virtual system instance details are displayed.

3. You can perform the following tasks with virtual system instances:
   - **Start an existing virtual system instance** Virtual system instances managed by IBM Cloud Orchestrator are not always running and in the started state. When a virtual system instance is in the stopped state, you can restart the virtual system instance to redeploy the virtual system instance into the cloud.
   - **“Stopping a persistent virtual system instance (classic)” on page 455.** Virtual system instances can be stopped without removing the virtual system instance from IBM Cloud Orchestrator. If a virtual system instance is stopped, then the virtual system instance is not running, but management of the virtual system instance is retained by IBM Cloud Orchestrator and the virtual system instance remains available for redeployment in the future.
   - **“Removing a virtual system instance (classic)” on page 456.** You can remove a virtual system instance when it is no longer needed. By removing a virtual system instance, you release all the IBM Cloud Orchestrator resources, making them available for placement decisions.
   - **“Creating a snapshot image” on page 457.** You can create a snapshot image to store the current state of the virtual system instance. You can later use this snapshot image to partially restore your virtual system instance to the stored state.
   - **“Restoring virtual system instances (classic) from a snapshot image” on page 458.** A snapshot image represents a previously captured state of the virtual system. Using this snapshot image, you can restore the state of virtual machines that were present in the virtual system instance to their stored state at the time the snapshot was taken.
   - **“Deleting snapshot images” on page 459.** You can delete a snapshot image of a virtual system instance that you no longer require.
   - **“Accessing virtual machines in your virtual system instance (classic)” on page 460.** Each virtual system instance consists of a set of virtual machines
that represent a physical node in an application server environment. You can access the individual virtual machines that make up your virtual system instance from the IBM Cloud Orchestrator user interface.

- **Viewing the details for your virtual machines** on page 462. Each virtual system instance consists of a set of virtual machines that represent a physical node in an application server environment. The details of each of these virtual machines can be viewed and monitored from the panel displaying the details for the virtual system instance.

**Results**

After you have followed these steps, your virtual system instance is ready to be used.

**Starting a persistent virtual system instance (classic):**

Virtual system instances managed by IBM Cloud Orchestrator are not always running and in the started state. When a persistent virtual system instance is in either the stopped state or the stored state, you can restart the virtual system instance to redeploy the virtual system instance into the cloud.

**Before you begin**

You must specifically be granted access to the virtual system instance you intend to start or be assigned the **admin** role to perform these steps. These steps are only intended for starting a virtual system instance that is in the stopped state or the stored state. To create a virtual system instance, you must deploy a pattern into the cloud. See **“Deploying a virtual system pattern (classic)” on page 440** for more information about creating a virtual system instance by deploying a pattern.

**About this task**

When a virtual system instance is stopped, the IBM Cloud Orchestrator resources are not released and the virtual system instance remains managed by IBM Cloud Orchestrator. The virtual system instance still has an impact on placement decisions though it is not actively running on the hypervisor. The IBM Cloud Orchestrator resources assigned to this virtual system instance are maintained to ensure that IBM Cloud Orchestrator resources are available when the virtual system instance is restarted.

If your virtual system instance has been stored, then other virtual system instances might have consumed the memory required to restart your virtual system instance. If this scenario occurs, then you can stop and then store other virtual system instances to release sufficient memory to ensure that your stored virtual system instance can be restarted. Follow these steps to redeploy the virtual system instance into the cloud by restarting the virtual system instance.

**Procedure**

From the Virtual System Instances window, click the start icon to deploy the virtual system instance into the cloud. Deployment of the virtual system instance into the cloud does not happen instantly. The deployment time depends on the virtual system instance size and the system activity. The starting icon is displayed while the deployment process is in progress or all the virtual machines in a cluster have not yet started. When the state of the virtual system instance is **The virtual system has been deployed and is ready to use**, then the virtual system instance is
running in the cloud and available for use. The failed icon is displayed if the
virtual system instance does not start successfully.

Results

Your virtual system instance is started and ready to be used.

What to do next

You can now access and use your virtual system instance. See “Accessing virtual
machines in your virtual system instance (classic)” on page 460 for more
information.

Stopping a persistent virtual system instance (classic):

You can stop a persistent virtual system instance without removing the virtual
system instance from IBM Cloud Orchestrator. If you stop a virtual system
instance, the virtual system instance is not running, but management of the virtual
system instance is retained by IBM Cloud Orchestrator and the virtual system
instance remains available for redeployment in the future.

Before you begin

You must specifically be granted write or all access to the virtual system instance or
be assigned the admin role to perform these steps.

About this task

When you stop a persistent virtual system instance, the resources are not released.
A stopped virtual system instance still affects placement decisions even though it is
not actively running on the hypervisor. The resources assigned to this virtual
system instance are maintained to ensure that the resources are available when you
redeploy the virtual system instance in to the cloud. Follow these steps to redeploy
the virtual system instance in to the cloud by starting the virtual system instance.

Procedure

From the Virtual System Instances window, click the stop icon to stop your virtual
system instance. Stopping the virtual system instance does not happen instantly.
When the state of the virtual system instance is Stopped, then the virtual system
instance has finished stopping. All virtual machines are stopped when a virtual
system instance is stopped. If you must stop only certain virtual machines, then
this can be achieved using the associated virtual machine actions. Stopping a
virtual system instance does not release the associated resources. When a virtual
system instance is stopped, clicking the start icon restarts the virtual system
instance using the resources it had reserved.

Results

Your virtual system instance is no longer running but remains available for
redeployment in the future.

What to do next

Create a virtual system instance by deploying a pattern or access a different virtual
system instance that is started. See “Deploying a virtual system pattern (classic)”
on page 440
Removing a virtual system instance (classic):

You can remove a virtual system instance when it is no longer needed. By removing a virtual system instance, you release all the cloud resources, making them available for placement decisions.

Before you begin

You must specifically be granted all access to the virtual system instance or be assigned the admin role to perform these steps.

About this task

When a virtual system instance is stopped, the cloud resources are not released. The processor usage and the memory allocation associated with the virtual system instance effects placement decisions made by IBM Cloud Orchestrator. Though the virtual system instance is not actively running, placement decisions are still effected. The cloud resources assigned to this virtual system instance are maintained to ensure that they are available if the virtual system instance is redeployed into the cloud. Deleting the virtual system instance releases the resources and the virtual system instance are no longer a factor in placement decisions. Follow these steps to remove the virtual system instance from IBM Cloud Orchestrator.

Procedure

1. From the Virtual System Instances window, click the remove icon to remove the virtual system instance and release the IBM Cloud Orchestrator resources.
   Clicking the remove icon displays a window requesting confirmation that this virtual system can be deleted.

2. In the confirmation dialog, specify what you want to delete.

   **Delete the virtual system instances history and log files as well.**
   When deleting a virtual system instance, you can delete history information and logs from that virtual system instance. To retain this information, ensure that the Delete the virtual system instances history and log files as well check box is not selected in the dialog box.

   If this virtual system instance contains any scripts that are run at virtual system instance deletion, the check box must be disabled. Otherwise, you cannot see the logs from the run of that script.

   **Note:** Scripts run at virtual system instance deletion are only run if the virtual system instance is running when it is deleted.

   **Ignore errors on delete.**
   When deleting a virtual system instance, you are also presented the option to ignore any errors that occur with the deletion. If you attempt to delete a virtual system instance and all associated virtual machines cannot be deleted, the delete fails. You can use the Ignore errors on delete option to force deletion of the virtual system instance.
CAUTION:
This option is helpful in specific situations only, so use this option with caution. You might know that the virtual machines cannot be deleted and you choose to clean them up manually, for example. Or, you might know that the server that is hosting the virtual machine is no longer available. Therefore, the delete would not occur because the errors would block the delete. You can use the Ignore errors on delete check box in these circumstances to force deletion of a virtual system instance, even if the virtual machines cannot be deleted.

3. Click OK to delete the virtual system instance with the parameters you specified.

Results
After you have followed these steps, the virtual system instance has been removed from the cloud.

What to do next
You can create a virtual system instance by deploying a pattern or you can access any virtual system instance that is already started. See "Deploying a virtual system pattern (classic)" on page 440 for more information about creating a virtual system instance by deploying a pattern. See "Accessing virtual machines in your virtual system instance (classic)" on page 460 for more information about accessing a virtual system instance.

Creating a snapshot image:
You can create a snapshot image to store the current state of the virtual system instance (classic). You can later use this snapshot image to partially restore your virtual system instance to the stored state.

Before you begin
You must be granted access to the virtual system instance or have the admin role to complete this task.

Note: The snapshot operation takes only snapshot of the disk image, without preserving memory state. This is consistent with OpenStack model of snapshots.

About this task
Using the snapshot function, you can store the state information for each of the virtual machines in the virtual system instance as it is running. You can use this snapshot image to restore these virtual machines in the virtual system instance to the states that existed when the snapshot was taken. Be aware of the following conditions:
- By restoring the virtual system instance using a snapshot image, the current state of the virtual system instance is lost.
- You can create only one snapshot image for each virtual machine.
- When you create a snapshot image for a virtual system instance that already has a snapshot image stored, the existing snapshot is removed.
- Any virtual machines that are added to the virtual system instance after the snapshot image is taken are still present after restoring the virtual system instance to its previously stored state.
Procedure
1. Click PATTERNS > Instances > Virtual System Instances (Classic).
2. Select the virtual system instance for which you want to create a snapshot.
3. Click Create to take a snapshot image of the current state of the selected virtual system instance. The virtual system instance status becomes Snapshooting until the snapshot image is completed. When the snapshot is successfully created, it is listed under the Create and the Restore buttons.
4. Optional: You can add a description for the snapshot image by clicking Snapshot Description and entering your text. Then click Enter to store the description.

Results
After completing these steps, you have a snapshot image available to restore the state of the virtual machines in the virtual system instance to their stored state. Any virtual machines that are added after the snapshot image is taken are unaffected by the restore operation.

What to do next
You can continue working with your virtual system instance, and if needed at some later time, you can restore your virtual system instance using the snapshot image.

Restoring virtual system instances (classic) from a snapshot image:
A snapshot image represents a previously captured state of the virtual system. Using this snapshot image, you can restore the state of virtual machines that were present in the virtual system instance to their stored state at the time the snapshot was taken.

Before you begin
You must be granted access to the virtual system instance or have the admin role to complete this task.

About this task
Using the snapshot restore function, you can restore the state information for each of the virtual machines in the virtual system instance to their state when the snapshot was taken. When you restore a virtual system instance by using a snapshot image, the current state of the virtual system instance is lost.

Any virtual machines in the virtual system instance that were added after the snapshot image was taken are still present and are unaffected when you restore the virtual system instance to its previous state.

Procedure
1. Click PATTERNS > Instances > Virtual System Instances (Classic).
2. Select the virtual system instance for which you want to restore by using its snapshot image.
3. Click the Restore icon to restore the virtual system instance to its previously captured state. The restore process does not take place instantly, and the virtual system instance is not usable while the virtual system instance is being
restored. After the virtual system instance is restored, it is automatically stopped and must be restarted manually.

Results

The states of the virtual machines that were part of the virtual system instance are restored to the same state as when the snapshot image was created. Virtual machines that were added after the snapshot image was taken are still present and are unaffected by this process.

What to do next

You can now access the virtual system instance you restored by using the snapshot image.

Deleting snapshot images:

You can delete a snapshot image of a virtual system instance (classic) that you no longer require.

Before you begin

You must be granted access to the virtual system instance or have the admin role to complete this task.

About this task

After creating a snapshot image for a virtual system instance, you can delete the snapshot when you no longer need it. An existing snapshot image of a virtual system instance is automatically deleted when a new snapshot image is created.

Procedure

1. Click PATTERNS > Instances > Virtual System Instances (Classic).
2. Select the virtual system instance for which you want to delete the snapshot image.
3. Click Delete next to the snapshot.

Results

The snapshot image is deleted from the system memory.

Virtual System Instances (classic) fields on the user interface:

You can view and work with the virtual system instances in the Virtual System Instances (classic) window.

Virtual System Instances (classic) fields

The following fields are displayed in a virtual system instance:

Created on

Specifies the date and time when the virtual system instance was created.

This field is automatically generated.
From pattern
   Specifies the pattern that was used to create this virtual system instance. This field is displayed as a link to the associated pattern.

Using Environment profile
   Specifies the environment profile, if one was used when creating this virtual machine, by providing a link to that environment profile. Clicking the link displays the details for that environment profile. If none was specified, then this field says **None provided**.

Current status
   Specifies the state of the virtual machine.

Updated on
   Specifies the last date and time when the virtual system instance was updated.

Access granted to
   The user who first deployed the virtual system instance is automatically granted all access to the virtual image as the owner. If you want additional users to access this virtual system instance, you must manually add the projects to which these users belong. See “User roles in IBM Cloud Orchestrator” on page 206 for more information about object level permissions.

Snapshot
   Includes links to any snapshot images that have been taken of this virtual system instance.

History
   Specifies the activity that has been performed on this virtual system instance.

Virtual Machines
   Lists the virtual machines that are included in this virtual system instance. If an environment profile was used, then the virtual machine name is provided by the user who provides the environment profile. Expand any virtual machine to display detailed information about that virtual machine. For more information about the virtual machine fields, see “Virtual machine panel fields on the Virtual System Instances (Classic) window” on page 464.

Comments
   Specifies optional information a user can append to a virtual system instance.

Accessing virtual machines in your virtual system instance (classic):

Each virtual system instance consists of a set of virtual machines that represent a physical node in an application server environment. You can access the individual virtual machines that make up your virtual system instance from the user interface.

Before you begin

You must specifically be granted access to the virtual system instance you intend to access. Optionally, you can be assigned the **admin** role to perform these steps.
About this task

With IBM Cloud Orchestrator, you can access any of the virtual machines that are contained by the virtual system instances.

Procedure

1. You can access virtual machines from the Virtual System Instances (Classic) window by clicking PATTERNS > Instances > Virtual System Instances (Classic) on the menu.
2. From the left panel of the Virtual System Instances (Classic) window, select the virtual system instance containing the virtual machine. Information about the virtual system instance is displayed in the right panel of the window.
3. From the right panel of the window, expand Virtual Machines by clicking the expand icon to view a list of the virtual machines that exists in the selected virtual system instance.
4. Expand the details for your virtual machine by clicking the expand icon next to the `<virtual_machine_name>` for the virtual machine you want to access. The number of virtual machines that exist for the virtual system instance is dependent on the pattern that was deployed to create it. From the list of the virtual machines included in this virtual system instance, you can view the CPU and the Memory currently being used by a virtual machine.
5. Access your virtual machine. Use one of the following procedure to access your virtual machine:
   - Click Login under the SSH column to open a new browser window and access the virtual machine using SSH. A prompt is displayed to enter user name and password.
   - Click VNC under the Consoles section to access your virtual machine using Virtual Network Computing (VNC). During pattern creation, the default setting is for your host operating system to be configured to accept VNC connections. VNC connections can be disabled by modifying the virtual machine properties during pattern creation.
   - Click WebSphere under the Consoles section to access the WebSphere Application Server administrative console on your virtual machine.

Important: To access your virtual machine using VNC or to access the WebSphere Application Server administrative console on your virtual machine, your virtual machine must be accessible from the machine you are accessing the user interface. If a firewall is preventing the connection on the required port, you must open this port to establish a connection. In addition to this, the DNS server must be correctly configured to resolve the virtual machine host name. If the DNS is not configured correctly, the ip-address and hostname fields must be present in the hosts file (/etc/hosts on Linux, and c:\WINDOWS\system32\drivers\etc\hosts for Windows. For more information about configuring DNS, see Configuring DNS for the Workload Deployer IP groups.

Results

After completing these steps, you have accessed your virtual machine from the Virtual System Instances (Classic) window of the user interface.
Expanding disk on deployed virtual machines:

Use the Default LINUX resize disk or Default WINDOWS resize disk script package to expand the disk of a deployed virtual machine.

About this task

After deploying a virtual machine, you can change the related flavor by editing the Flavor field in the Virtual machines section in the Virtual System Instances (Classic) window. When you change the flavor only the vCPU and the memory values are changed. To change the disk size of a deployed virtual machine, you must follow one of the following procedures:

- If you added the Default LINUX resize disk or Default WINDOWS resize disk script package to the virtual system pattern before deploying it, perform the following steps:
  1. Click PATTERNS > Instances > Virtual System Instances (Classic) to access the Virtual System Instances (Classic) window.
  2. Select your instance and expand the Virtual Machines detailed section.
  3. In the Script Packages section, select the Default LINUX resize disk or Default WINDOWS resize disk script package and click Execute now.

For information about adding a script package to a virtual system pattern, see "Associating a script package with a pattern" on page 361.

- If the Default LINUX resize disk or Default WINDOWS resize disk script package is not part of the virtual system pattern, you must download the script package and run it manually by performing the following steps:
  1. Click PATTERNS > Pattern Design > Script Packages to open the Script Packages window.
  2. Select the Default LINUX resize disk or Default WINDOWS resize disk script package and click Download in the Script package file field in the right pane.
  3. When the file download is completed, upload the defaultlinuxresizedisk.zip or defaultwindowsresizedisk.zip script package zip file to the virtual machine where you want to change the disk size.
  4. Unzip the script package file and run the resize script.

Viewing the details for your virtual machines:

Each virtual system instance (classic) consists of a set of virtual machines that represent a physical node in an application server environment. These virtual machines are assigned to and hosted by a hypervisor. You can monitor the details of each of these virtual machines.

Before you begin

You must specifically be granted access to the virtual system instance you intend to access. Optionally, you can be assigned the admin role to perform these steps.

If the virtual machines are running on a VMware hypervisor, the VMware tools must be installed in the operating system.
About this task

You can view and monitor the details about each virtual machine in your virtual system instance by following these steps.

The values are updated frequently and the user interface must be refreshed to get the newest data.

Procedure

1. From the left panel of the Virtual System Instances (Classic) window, select the associated virtual system instance in the left panel. Information about the virtual system instance is displayed in the right panel of the window.

2. From the right panel of the window, expand **Virtual Machines** by clicking the expand icon to view a list of the virtual machines that exists on the selected virtual system instance. The number of virtual machines that exist for a virtual system instance is dependent on the pattern that was deployed. Some information is displayed for each virtual machine without having to expand to see additional details. From the list of virtual machines included in this virtual system instance, you can view the following details for each virtual machine.

   **CPU**  This field graphically specifies the percentage of the virtual CPU power that is currently being used. The number of virtual CPUs available is determined by the pattern used to create the virtual system. The default number of virtual CPUs for a virtual machine is one.

   **Memory**  This field displays a graph that specifies the percentage of the memory that is currently being used by the virtual machine. The amount of memory available is determined by the pattern used to create the virtual system instance. The default amount of virtual memory for a virtual machine is 2048 MB.

   **SSH**  This field provides the **Login** link that you can click to log in to your virtual machine using Secure Shell (SSH). You are prompted to log on as a user of your choosing.

   **Actions**  This field displays the available actions for a virtual machine. Actions that are not available for a specific virtual machine are not active. Click the **View** link to show or hide the available actions for the virtual machine.

   **Clone**  Deploys a new instance of the virtual machine with the same parameters. Script packages marked to run at **virtual system creation** run after the virtual machine is created. Any maintenance applied to the source virtual machine is applied to the cloned virtual machine automatically. The new virtual machine does not inherit any disk changes made to the source virtual machine. This action is available if you are viewing a virtual machine that has completed deployment and is ready to be cloned.

   **Note:** The tooltip **This resource is not ready to be cloned** is shown for images that cannot be cloned temporary or permanently. It actually means **This resource cannot be cloned**.

   **Start**  Restarts a virtual machine that has been stopped.
Stop  Stops a virtual machine that has been started.
Delete  Deletes a virtual machine after it is stopped. Resources associated with the virtual machine are released.

3. Expand the details for a virtual machine by clicking the expand icon next to the `<virtual_machine_name>`.

4. View your virtual machine details. For information about the fields shown for each virtual machine, see “Virtual machine panel fields on the Virtual System Instances (Classic) window.”

Virtual machine panel fields on the Virtual System Instances (Classic) window:

You can view information about the virtual machines that are associated with a virtual system instance using the Virtual machines panel on the Virtual System Instances (Classic) window of the user interface.

Virtual machine summary fields

The Virtual machines section, before being expanded, shows a status line. The status line shows the total number of virtual machines, and the number of virtual machines in each state. For example, the status bar could read: 4 total - 1 deleted - 3 started - 1 failed to indicate the total number of virtual machines and how many are in which state. When expanded, the Virtual machines section lists the managed virtual machines that IBM Cloud Orchestrator started for the workload.

When the Virtual machines section is expanded, the following fields are provided for each virtual machine listing, without expanding the details for each virtual machine:

Name  The name of the virtual machine that is included in the virtual system being viewed. If an environment profile was used, then the virtual machine name could have been provided by the user who provided the environment profile.
CPU  Provides the graphic and numeric percentage of available CPU being used by this virtual machine.
Memory  Provides the graphic and numeric percentage of available memory being used by this virtual machine.
SSH  Clicking the Login link opens a dialog to log in to an SSH session.
Actions  Clicking the View link displays or hides the following set of icons to work with this virtual machine:

Clone  Deploys a new instance of the virtual machine with the same parameters. Script packages marked to run at virtual system creation run after the virtual machine is created. Any maintenance applied to the source virtual machine is applied to the cloned virtual machine automatically. The new virtual machine does not inherit any disk changes made to the source virtual machine. This action is available if you are viewing a virtual machine that has completed deployment and is ready to be cloned.

Note: The tooltip This resource is not ready to be cloned is shown for images that cannot be cloned temporary or permanently. It actually means This resource cannot be cloned.
**Start**  
Restarts a virtual machine that has been stopped.

**Stop**  
Stops a virtual machine that has been started.

**Delete**  
Deletes a virtual machine after it is stopped. Resources associated with the virtual machine are released.

**Check box**  
Checking the check box in the header of this column selects all of the virtual machines that have not been deleted. Clearing the check box in the header row also clears all of the virtual machines in the listing. Use this check box with the **Group Actions** link to apply an action to the selected group of virtual machines.

**Group Actions**  
Select this check box to apply an action to the group of virtual machines selected. Actions can be applied to all of the virtual machines that have not been deleted or to a selected group of virtual machines.

**General information**

When the view of any individual virtual machine is expanded, the **General information** section shows the following information about that virtual machine:

**Created on**  
Specifies the time and date the virtual machine was created. This field is automatically generated for managed virtual machines.

**From virtual image**  
Specifies the virtual image that was used when creating this virtual machine by providing a link to that virtual image in the catalog. Clicking the link displays the details for that virtual image.

**Note:** After applying a service pack, the templateid of the virtual machine might be updated. For example, this templateid might be updated from 70017 to 70019. This is an expected behavior.

**Part name**  
The name of the part. This field is not available for unmanaged virtual machines.

**Current status**  
Specifies the status of the virtual machine.

**Updated on**  
Specifies the time and date of the last change to the virtual machine.

**On hypervisor**  
Specifies the hypervisor where this virtual machine is located by providing a link to the hypervisor details panel. You can click the link to display the details of the hypervisor where this virtual machine is running.

**In cloud group**  
Specifies the cloud group where this virtual machine is located by providing a link to the Cloud Groups panel. You can click the link to display the details of the cloud group where this virtual machine is running.

**Registered as**  
Specifies how the virtual machine is registered on the hypervisor.
Stored on
Specifies the storage device associated with this virtual machine.

Hardware and network

The Hardware and network section provides the following fields:

**Flavor** Specifies the instance flavor that describes the memory and storage capacity of the virtual machine. By default the flavor values are:

- **m1.tiny**
  Memory: 512 MB, vCPUs: 1, Storage: 0 GB
- **m1.small**
  Memory: 2048 MB, vCPUs: 1, Storage: 20 GB
- **m1.medium**
  Memory: 4096 MB, vCPUs: 2, Storage: 40 GB
- **m1.large**
  Memory: 8192 MB, vCPUs: 4, Storage: 80 GB
- **m1.xlarge**
  Memory: 16384 MB, vCPUs: 8, Storage: 160 GB

To change the flavor, stop the virtual machine and click **Edit** to select a new flavor value. When you confirm your change, the virtual machine is automatically restarted.

**Note:** When you change the flavor only the vCPU and the memory values are changed. To change the disk size of a deployed virtual machine, you must follow the procedure described in “Expanding disk on deployed virtual machines” on page 462.

**Note:** If you are using IBM Cloud Orchestrator in languages other than English, the **Flavor** field might be displayed as **Untranslated message RM11388**.

**Virtual CPU count**
Specifies the number of CPU this virtual machine represents. This value is specified in the pattern that was deployed to create the virtual system instance. See “Working with virtual system patterns (classic)” on page 411 for more details on the pattern options.

**Physical CPU**
Specifies whether the physical CPUs are reserved for exclusive use by this virtual machine. The **Reserved** status is shown if the **True** value is selected for **Reserve physical CPUs** during deployment and it is an image variable.

**CPU shares on host**
The amount of CPU allocated for the host.

**CPU shares consumed on host**
The amount of CPU actually used by the host.

**Virtual memory (MB)**
Specifies the IP address and the host name of this virtual machine. The virtual machine must be stopped before this value can be changed.

**SSH public key**
The name of, and a link to, the public key.
Network interface
The network interface address.

MAC address
Specifies the Media Access Control (MAC) address of the virtual machine.

Operating System
The Operating System section provides the following fields:

Name
Specifies the type of operating system that is running on the virtual machine.

Type
Shows the specific variety of the operating system that is running on the virtual machine.

Version
Specifies the equivalent of the `uname -a` command for the operating system on the virtual machine.

Note: After applying a service pack, the version of the operating system might not be displayed by the `uname -a` command. The initial value is obtained from the virtual machine, which is then stored in the database. This is expected behavior.

WebSphere Configuration
The WebSphere Configuration section provides the following fields:

Cell Name
The cell in which this virtual machine resides.

Node Name
The node in which this virtual machine resides.

Profile Name
The profile in which this virtual machine was run.

Show all environment variables
This link specifies the set of supplied environment variables that you can use when using a script package. See “Script package environment variables” on page 365 for more information about the environment variables.

Script Packages
The Script Packages section lists any script packages that have been run on this virtual machine. If any script packages have been run for this virtual machine, then links to the associated log files are also included. See “Managing script packages” on page 354 for more information about script packages.

If a script is user initiated, meaning the `executes` attribute is set as when I initiate it, then the start icon is displayed next to the script name. Click the icon to run the script. There is no limit on the number of times a script is run using this method.

Scripts in this section can include add-ons. For more information about add-on scripts, see “Managing add-ons” on page 385.
Consoles

The Consoles section provides a link to access your virtual machine. Using the provided links, you can access the WebSphere Application Server administrative console for your virtual machine. See "Accessing virtual machines in your virtual system instance (classic)" on page 460 for more information about accessing your virtual machines.

Managing pattern types

A virtual application pattern type is a collection of plug-ins that identify components, links and policies, along with configuration files, packaged in a .tgz file. The virtual application patterns are used to build a virtual application that includes these components, links and policies.

About this task

The following pattern type is shipped with IBM Cloud Orchestrator:

- IBM Foundation Pattern. It provides shared services for deployed virtual applications such as monitoring and load balancing.

Note: The Foundation Pattern is a prerequisite to using all other pattern types. Other pattern types are purchasable at the IBM PureSystems Centre.

Note: The following pattern types are supported by the current version of IBM Cloud Orchestrator:

- foundation 2.1.0.0
- dbaas 1.1.2.0
- webApp 1.0 1.0.1.0
- webApp 2.0 2.0.2.0
- application 1.0.2.0

If you created virtual applications that contain these pattern types, ensure to download the supported version of the pattern types to continue to use your virtual applications in IBM Cloud Orchestrator.

Use the following steps to work with pattern types:

Procedure

1. View the pattern types
2. View the system plug-ins in the pattern types
3. Import a pattern type
4. Accept the pattern license agreement. You must accept the license agreement for each pattern type that you want to use. You can accept the license by following these steps:
   a. Click PATTERNS > Pattern Types. The Pattern Types palette displays and the pattern types are listed.
   b. Select a pattern_type. The pattern details display on the right.
   c. In the License Agreement field, click Accept. The dialogue window displays for the pattern type.
   d. After reading the license agreement, click Accept. The license agreement details changes to Accepted.
   e. Click Enable to change the pattern type status to Available.
Important: By default, there is not a license agreement to accept for the Foundation Pattern type. But, the Foundation Pattern must be made available before the other pattern types can be made available and used.

For detailed information about accepting the license agreement for specific pattern types, see the related pattern type documentation.

5. Upgrade a pattern type
6. Remove a pattern type

Results

You are ready to start creating or extending virtual applications with pattern types.

Viewing pattern types

The system includes a set of pattern types that you can use to create solution-specific virtual applications.

Before you begin

The IBM Foundation Pattern type is shipped with IBM Cloud Orchestrator. Other pattern types are purchasable at the IBM PureSystems Centre.

By default, there is not a license agreement to accept for the foundation pattern type. However, the foundation pattern type must be enabled before the other pattern types can be enabled. The foundation pattern type is a prerequisite to using all other pattern types.

About this task

You can view information about the pattern type in the user interface by following these steps:

Procedure

1. Click PATTERNS > Deployer Configuration > Pattern Types. The Pattern Types palette displays and the pattern types are listed.
2. Select a pattern_type. The pattern details display on the right.
3. View the details of the pattern type, including:
   - **Description**
     Specifies the description of the pattern type.
   - **License agreement**
     Specifies if the license agreement is accepted.
   - **Status**
     Specifies the status of the pattern type: Disabled or Available. To enable the pattern type, select Enable. After you enable the pattern type, the status is changed to Available. You can either enable the current pattern type of no dependencies exist, or enable all of the pre-requirements, such as accepting licenses and updating statuses.
   - **Required**
     Specifies any prerequisite patterns that are required.
   - **Plug-ins**
     Specifies the plug-ins that are associated with this pattern type. Click
show me all plug-ins in this pattern type to view plug-ins associated with the pattern type. Plug-ins required for configuration are also listed.

Dependency
Lists pattern type dependencies.

Viewing the plug-ins in the pattern types
You can view the system plug-ins that are associated with the pattern types.

Procedure
1. Click PATTERNS > Deployer Configuration > Pattern Types. The Pattern Types palette displays and the pattern types are listed.
2. Select a pattern type. The pattern details display on the right.
3. Click show me all plug-ins in this pattern type and the System Plug-ins palette displays with a list of plug-ins.

Results
You have viewed the plug-ins that are associated with the pattern type.

Importing pattern types
You can import a new pattern type to IBM Cloud Orchestrator.

Before you begin
You must be assigned the admin role to perform these steps.

Note: To upload a file that is larger than 2 GB, you must upload it from a remote system. To upload a file from a local system, the size of the file must be smaller than 2 GB.

About this task
Complete the following procedure to upload and import a new pattern type from a local system.

Procedure
1. Click PATTERNS > Deployer Configuration > Pattern Types. The Pattern Types palette is displayed.
2. Click the New icon. The Install a pattern type window is displayed.
3. Select the Local tab, and click Browse to select the .tgz file to import as a pattern type. Your system proceeds to upload the .tgz file.

Results
You have imported a new pattern type.

What to do next
Now you must accept the license agreement of the pattern type, configure plug-ins in this pattern type, and enable the pattern type if you want to use it.
Removing a pattern type
You can remove a pattern type from the IBM Cloud Orchestrator.

Before you begin
You might not be able to delete a pattern type if one or more of the following is true:
• The pattern type is in use, for example, the pattern type is associated with deployed an application.
• The pattern type is a prerequisite of another pattern type, such as the IBM Foundation Pattern.
• One or more plug-ins of this pattern type is in use, for example, the plug-ins are associated with a deployed application.

About this task
Procedure
1. Click PATTERNS > Deployer Configuration > Pattern Types. The Pattern Types palette displays.
2. Select the pattern type to delete.
3. Click the Delete this pattern type icon. The Do you really want to delete this pattern type window displays.
4. Click OK to delete the pattern type.

Results
You have removed a pattern type.

Upgrading pattern types
Periodic updates are available for pattern types.

Before you begin
The pattern type updates are available at IBM Fix Central.

About this task
Pattern types are packaged in a .tgz file, whether the delivery is in release, update or fix pack. For example, a web application pattern update looks like the following: webapp-x.x.x.x.tgz (release), webapp-x.x.x.x.tgz (update on Fix Central) or webapp-x.x.x.x.tgz (update through a fix pack on Fix Central), where x.x.x.x is the release level.

If you download an update from Fix Central, and import the file into IBM Cloud Orchestrator, the administrator must accept the license agreement and make the version.release (VR) available. If you download a fix pack that includes updated pattern type and import the file into IBM Cloud Orchestrator, the new pattern type license is already accepted and the pattern type is automatically available to the user.

The following guide explains how various users are impacted by pattern type updates:
• The user is exposed to pattern types at VR when they create a workload.
The administrator can import and delete pattern types at version.release.minor.fixpack (VRMF).

The administrator manages plug-ins in pattern types at VR. The administrator does not need to know which VRMF contains the latest of plug-ins that are used in the VR.

New workloads are based on the latest plug-ins within that VR; locked applications and existing deployments are not affected.

For an update, the administrator must accept the license agreement and make the VR available.

Users see a new pattern type when creating a workload (pattern type at VR).

After you download the fixes from Fix Central, you can import the new pattern type by performing the following steps:

Procedure
1. Click PATTERNS > Deployer Configuration > Pattern Types. The Pattern Types palette displays.
2. Click the New icon. The Install a new pattern type window displays.
3. Click Browse to select the .tgz file to import as a pattern type. Your system proceeds to upload the .tgz file.

Results
You have imported an updated .tgz file as an upgraded pattern type.

Upgrading a deployed pattern type
You can upgrade a pattern type that is associated with a virtual application instance through the IBM Cloud Orchestrator user interface.

Before you begin
The pattern type license must be enabled before you can upgrade. Click PATTERNS > Deployer Configuration > Pattern Types > pattern_type > Enable.

About this task
Perform the following steps:

Procedure
1. View the virtual application instances. Click PATTERNS > Instances > Virtual Application Instances.
2. Select the virtual_application_instance for which you want to upgrade the pattern type. Click the Upgrade icon. A dialogue box displays where you can confirm that you want to upgrade the pattern type to the latest version. Click OK. A message displays across the menu that the pattern type is upgrading. When the upgrade is complete, the Status field arrow turns green.
Pattern type packaging reference

The pattern types are a collection of plug-ins. The plug-ins contain the components, policies and links of the virtual application pattern. This topic explains the packaging of the plug-ins that create the virtual application pattern type.

Virtual application pattern types are shipped with IBM Cloud Orchestrator or they are purchasable at the [IBM PureSystems Centre](#).

The associated plug-in files that are associated with a pattern type are as follows:

- `{ptype}.tgz` file
- `plugins/set of {plugin}.tgz` files
- `files/set of {name}` files

The `{ptype}.tgz` file is required and must contain the `patterntype.json` file. The `{ptype}.tgz` file might also contain the license and localized messages. For example, the `patterntype.json` file for the IBM Web Application Pattern (not released with the product) is as follows:

```
{
    "name": "NAME",
    "shortname": "webapp",
    "version": "2.0.0.0",
    "description": "DESCRIPTION",
    "prereqs": {
        "foundation": "*"
    },
    "license": {
        "pid": "5725057",
        "type": "PVU"
    }
}
```

A pattern type defines a logical collection of plug-ins, but not the members. The members (plug-ins) define their associations with pattern types in the `config.json` file. Therefore, pattern types are dynamic collections and can be extended by third parties. For example, the `config.json` file for the DB2 plug-in (not released with the product) is as follows:

```
{
    "name": "db2",
    "version": "2.0.0.0",
    "files": [
        "db2/db2_wse_en-9.7.0.3a-linuxx64-20110330.tgz",
        "optim/dsadm223_iwd_20110420_1600_win.zip",
        "optim/dsdev221_iwd_20110421_1200_win.zip",
        "optim/com.ibm.optim.currency.administrator.pek_2.2.jar",
        "optim/com.ibm.optim.development.studio.pek_2.2.jar"
    ],
    "patterntypes": {
        "primary": {
            "dbaas": "1.0"
        },
        "secondary": [
            {
                "webapp": "2.0"
            }
        ]
    },
    "packages": {
        "DB2": [
            {
                "persistent": true,
            }
        ]
    }
}
```

Chapter 7. Managing and deploying virtual patterns 473
Working with virtual applications

IBM Cloud Orchestrator provides support for creating application-centric deployments called virtual applications. Virtual application builders and deployers describe virtual applications in terms of the application artifacts and required quality of service levels. IBM Cloud Orchestrator determines the infrastructure and middleware that is required to host the virtual application at deployment time.

Virtual application patterns

IBM Cloud Orchestrator provides a generic framework for designing, deploying, and managing virtual applications. The model that you build by using the application artifacts and quality of service levels is called a virtual application pattern. You can use predefined patterns, extend existing patterns, or create new ones.

When you build a virtual application pattern, you create the model of a virtual application by using components, links, and policies.

Consider an order management application with the following requirements:

- It is web-based and runs on WebSphere Application Server.
- It is highly available, with a cluster of two WebSphere Application Server nodes and two DB2 nodes.
- It uses DB2 to store order information and other application data.
- It supports the current maximum number of orders that are received by the company, 10 transactions per second, but it must also be able to scale to handle a larger number of transactions, up to 15 transactions per second, as the business increases.
- It supports backup and restore capabilities.

A virtual application builder can use IBM Cloud Orchestrator to create a virtual application pattern by using components, links, and policies to specify each of these parameters.

Component

Represents an application artifact such as a WAR file, and attributes such as a maximum transaction timeout. In terms of the order management application example, the components for the application are the WebSphere Application Server nodes and the DB2 nodes. The WebSphere Application Server
Server components include the WAR file for the application, and the DB2 components connect the application to the existing DB2 server.

**Link**  A connection between two components. For example, if a web application component has a dependency on a database component, an outgoing link from the web application component to the database component defines this dependency. In terms of the order management application example, links exist between the WebSphere Application Server components and the DB2 components.

**Policy**  Represents a quality of service level for application artifacts in the virtual application. Policies can be applied globally at the application level or specified for individual components. For example, a logging policy defines logging settings and a scaling policy defines criteria for dynamically adding or removing resources from the virtual application. In terms of the order management application example, a Response Time Based scaling policy is applied that scales the virtual application in or out to keep the web response time 1000 - 5000 ms.

When you deploy a virtual application, the virtual application pattern is converted from a logical model to a topology of virtual machines that are deployed to the cloud. Behind the scenes, the system determines the underlying infrastructure and middleware that is required for the application, and adjusts them as needed to ensure that the quality of service levels that are set for the application are maintained. A deployed topology that is based on a virtual application pattern is called a *virtual application instance*. You can deploy multiple virtual application instances from a single virtual application pattern.

The components, links, and policies that are available to design a particular virtual application pattern are dependent on the *pattern type* that you choose and the *plug-ins* that are associated with the pattern type.

**Virtual application pattern types and plug-ins**

A *pattern type* represents a collection of related components, links, and policies that are used to build a set of virtual applications. A pattern type defines a virtual application domain. For example, the IBM Web Application Pattern pattern type (not released with the product) defines a domain in which J2EE web applications are deployed. It includes components for WAR, EAR, and OSGiEBA files. These components have an attribute for the appropriate archive file, which an application builder specifies during construction of the virtual application pattern.

The web application can connect to a database, so the pattern type also includes a component to represent the database and provides its connection properties as attributes. The pattern type also defines a link between the database and the WAR file to represent communication between the application and the database.

The application components (WAR, EAR, OSGiEBA) can all be configured with quality of service levels by applying policies. The available options include scaling, routing, logging, and JVM policies.

The plug-ins that are associated with Web Application Pattern define these components, links, and policies. They also provide the underlying implementation to deploy the virtual applications in the cloud and perform maintenance on deployed virtual application instances.
Virtual application builders create virtual application patterns in the Pattern Builder. Within Pattern Builder, you begin by selecting the pattern type that you want to use. This choice determines the set of components, links, and policies that you can use to build the virtual application and the type of virtual applications that you can create.

Plug-in developers are responsible for creating or customizing pattern types and plug-ins that control the available components, links, and policies and corresponding configuration options, as well as the code for implementing deployments.

**Options for reusing virtual application configuration**

To simplify complex application design and standardize reuse of common components and configuration across multiple virtual applications, you can leverage several options:

**virtual application templates**
A predefined virtual application pattern that can include components that are already pre-configured. You can use a template as a foundation for creating virtual application patterns that use the same basic configuration. Alternatively, you can deploy a virtual application directly from a template and specify any required settings at deployment time.

**virtual application layers**
A grouping of components within a virtual application pattern. You can set up multiple layers within a single virtual application pattern or you can import one virtual application pattern into another as a reference layer.

**Reusable components**
You can pre-configure a virtual application component and save it for reuse by any virtual application pattern based on the same pattern type.

**Creating a virtual application pattern**

Complete the following steps to get started with creating a virtual application pattern.

1. Design your application. You can create a new virtual application pattern, or create one based on an existing template.
   - Create a virtual application pattern
     Virtual application patterns are fully configured before you deploy them.
   - Create a virtual application template
     Virtual application templates are designed for more flexibility. Properties can be left unconfigured or configured with default values and users who deploy virtual applications can specify the required values at deployment time.

2. Link components and configure them.
3. Use layers and reusable components to simplify creation of additional virtual application patterns.
   - Groups components into layers or import a virtual application into your virtual application pattern a layer.
   - Add predefined reusable components to the virtual application. For more information, see Working with reusable application components.
Virtual application pattern components

A virtual application pattern contains components that represent middleware services that are required by the virtual appliance instance.

You can connect components in a virtual application pattern to indicate dependencies and optionally apply a policy to configure middleware services during deployment to configure specific behavior or define a quality of service level. Components, links, and policies can have required and optional attributes.

Components, links, and policies are defined by plug-ins. When you create a virtual application pattern, the available components, links, policies, and configuration options are determined by the plug-ins that are included with the selected pattern type.

Components

The following components are available with the virtual application patterns provided with IBM Cloud Orchestrator or purchasable at the IBM PureSystems Centre.

Important: Components that require disk add-ons are not supported in IBM Cloud Orchestrator.

- Application
  - Additional archive file
  - Enterprise application
  - “Existing web service provider endpoint” on page 501
  - “Policy set” on page 502
  - Web application such as IBM WebSphere Application Server
- Database
  - Data Studio web console
  - Database such as IBM DB2
  - “Existing database (DB2)” on page 512
  - Existing database (Informix)
  - Existing database (Oracle)
  - “Existing IMS database” on page 517
- Messaging
  - “Existing Messaging Service (WebSphere MQ)” on page 529
  - “Existing queue (WebSphere MQ)” on page 532
  - “Existing topic (WebSphere MQ)” on page 531
- OSGi
  - External OSGi bundle repository
  - OSGi application
- Transaction Processing
  - CICS® Transaction Gateway
  - “Existing IMS TM” on page 541
- User Registry
  - “Existing User Registry (IBM Tivoli Directory Server)” on page 518
  - “Existing User Registry (Microsoft Active Directory)” on page 522
  - “User Registry (Tivoli Directory Server)” on page 525
Other components
- Generic target
- "Debug" on page 585

Policies

You can optionally apply policies to a virtual application to configure specific behavior in the deployed virtual application instance. Two virtual applications might include identical components, but require different policies to achieve different service level agreements. For example, if you want a web application to be highly available, you can add a scaling policy to the web application component and specify requirements such as a processor usage threshold to trigger scaling of the web application. At deployment time, the topology of the virtual application is configured to dynamically scale the web application. Multiple WebSphere Application Server instances are deployed initially for the web application and instances are added and removed automatically based on the service levels that are defined in the policy.

Policies can be applied only to particular types of components. For more information, see the following links:
- Scaling policy
- Routing policy
- Java virtual machine (JVM) policy
- Log policy

Managing virtual applications

A virtual application is defined by a virtual application pattern. It is a complete set of platform resources that fulfill a business need, including web applications, databases, user registries, messaging services and transaction processes. The pattern used to create the virtual application is a collection of plug-ins that provide these resources and services in the form of components, links and policies.

Before you begin

To manage virtual applications, you must have the catalogeditor role or the admin role. You must also accept the license agreements for the pattern types in your environment before you can use the pattern type to create and extend virtual applications.

Note: The base image for virtual applications must be downloaded from Passport Advantage. Its name is IBM OS Image for Red Hat Linux Systems. We recommend always to get the latest available version.

About this task

The plug-in component, links and policies are selected when you create or extend your virtual application in the Pattern Builder. You can find information about plug-in components, links, and policies at "Virtual application pattern components" on page 477. You can start with a virtual application template of an application. The virtual application is deployed to the cloud and becomes a virtual application instance.
Creating virtual application patterns

Create virtual application patterns to model virtual applications that you can deploy to the cloud.

Before you begin

You must be assigned the catalogeditor role or the admin role.

Note: The base image for virtual applications must be downloaded from Passport Advantage. Its name is IBM OS Image for Red Hat Linux Systems. We recommend always to get the latest available version.

About this task

When you create a virtual application, you first select a pattern type. The pattern type abstracts the infrastructure and middleware layers for a particular type of workload, such as a web application. You can then create a virtual application pattern by using the components associated with the selected pattern type.

Use the Pattern Builder to define, create, and deploy your virtual applications. For example, rather than installing, configuring, and creating a connection to a specific instance of a database, you can specify the need for a database and provide the associated database schema in your virtual application pattern. The database instance and the connection in the cloud is then created for you by the workload pattern.

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns. The Virtual Application Patterns palette displays.
2. Click the Add icon on the toolbar.
3. To build your virtual application pattern:
   a. Select a pattern type from the drop-down menu.
   b. Select a virtual application template.
   c. Click Start Building. You have created a new virtual application associated with a pattern type. The Pattern Builder opens in a new window where you can add components and policies.
4. On the Virtual Application properties pane, specify the following information:
   - Name: The name of the virtual application pattern.
   - Description: (Optional) The description of the virtual application pattern.
   - Type: Leave Application selected to create a virtual application pattern or select Template to create a virtual application template that is used as the basis for creating other virtual application patterns.
   - Lock option for plugin usage: Specify how this virtual application pattern is affected by upgrades to the pattern type or to IBM Foundation Pattern.
      - Unlock plugins: If the pattern type is upgraded use the latest versions of pattern type plug-ins. If IBM Foundation Pattern is upgraded, use the latest version.
Lock all plugins
Do not change the version of plug-ins or the version of the IBM Foundation Pattern associated with this virtual application pattern when an upgrade occurs.

Lock all plugins except Foundation plugins
If the pattern type is upgraded, do not change the version of the plug-ins associated with this virtual application pattern. If IBM Foundation Pattern is upgraded, use the latest version.

Note: You can view a list of which plug-ins are locked if you select Lock all plugins or Lock all plugins except Foundation plugins. Click the Source tab in Pattern Builder. The application model source is displayed. Search for the element plugins to view the list.

5. If you selected a blank template, the canvas is empty and you can start building the virtual application. If you selected a template, customize the virtual application:
   • Drag the components that you want to add to the virtual application pattern onto the canvas.
   • Use reusable components that you created. For more information, see Working with reusable application components. A reusable component is a saved configuration that can be reused to build applications. Components that are reusable display a triangle. Click the triangle to get a list of reusable components. Then drag the reusable component to the Pattern Builder canvas. This creates a new component that has the same configuration as the reusable component. You can also add a reusable component in Pattern Builder. Select a component in the canvas, configure its properties and then click Add to my palette. The reusable component displays in the left pane.
   • To add policies to the virtual application pattern, click Add policy for application and select a policy or select a component part on the canvas and click the Add a Component Policy icon to add a component-specific policy.
   • To remove parts, click the Remove Component icon in the component part.
   • To edit the connections between the parts, hover over one of the objects until the blue circle turns orange. Select the circle with the left mouse button, drag a connection to the second object until the object is highlighted, and release the mouse button.

6. Click Save.

Results
The virtual application pattern is created.

Editing virtual application patterns
You can edit a virtual application pattern to add or remove application components, links and policies.

Before you begin
You must be granted access to the pattern or assigned the admin role.

About this task
Use the Pattern Builder to edit your virtual application pattern. You can add, edit or remove components, links and policies.
**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**. The Virtual Application Patterns palette displays.
2. Select a virtual application pattern and click the **Open** icon.
3. Edit the virtual application pattern, as required:
   - Drag the components that you want to add to the virtual application pattern onto the canvas.
   - **Use reusable components that you created.** For more information, see [Working with reusable application components](#). A reusable component is a saved configuration that can be reused to build applications. Components that are reusable display a triangle. Click the triangle to get a list of reusable components. Then drag the reusable component to the Pattern Builder canvas. This creates a new component that has the same configuration as the reusable component. You can also add a reusable component in Pattern Builder. Select a component in the canvas, configure its properties and then click **Add to my palette**. The reusable component displays in the left pane.
   - To add policies to the virtual application pattern, click **Add policy for application** and select a policy or select a component part on the canvas and click the **Add a Component Policy** icon to add a component-specific policy.
   - To remove parts, click the **Remove Component** icon in the component part.
   - To edit the connections between the parts, hover over one of the objects until the blue circle turns orange. Select the circle with the left mouse button, drag a connection to the second object until the object is highlighted, and release the mouse button.
4. Click **Save**.

**What to do next**

Deploy the virtual application.

**Cloning virtual application patterns**

Use the console to clone an existing virtual application pattern.

**Before you begin**

You must be granted access to the pattern and be assigned the **catalogeditor** role or the **admin** role.

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**.
2. Select a virtual application pattern and click the **Clone** icon on the toolbar.
3. Specify the name for the copy of the virtual application pattern and click **OK**.

**What to do next**

Edit the cloned virtual application pattern as necessary.
Deleting virtual application patterns
You can delete virtual application patterns that you no longer need to deploy.

Before you begin
You must be granted access to the application pattern or assigned the admin role.

Procedure
1. Click PATTERNs > Pattern Design > Virtual Application Patterns.
2. Select a virtual application pattern. The virtual application pattern details display in the right pane.
3. Click the Delete icon on the toolbar.
4. Click Confirm to confirm that you want to delete the pattern.

Results
You deleted a virtual application pattern.

Creating virtual application layers
You can use the Pattern Builder in IBM Cloud Orchestrator to create virtual application layers, which provides a way for you to control the complexity and reuse virtual applications.

About this task
By default, a virtual application consists of one layer when you first create it. When you use application layering, you can modify an existing virtual application by adding separate layers. For example, you can use one virtual application that defines the basic components of one deployment environment to create a different virtual application for other deployment environments by associating the quality of service (QoS) layer with different QoS goals.

One virtual application can contain multiple layers. A layer can contain component types of the virtual application, or the layer can reference another virtual application, which is called a reference layer.

Because there is no predefined set of layers or binding between a component type and a particular layer, you can create layers according to your business goals. However, one component type in a virtual application can be placed in only one layer, but you can move parts between the layers.

You use the Pattern Builder to add or delete a layer, edit a layer, disable or enable a layer, move virtual application parts between layers, or import a virtual application as a reference layer.

Follow these steps to create an application layer:

Procedure
1. Click PATTERNs > Pattern Design > Virtual Application Patterns. The Virtual Application Pattern palette displays.
2. Select the virtual application for which you want to create a layer.
3. Click the Open icon to edit the pattern layer. The Pattern Builder displays. The Layers palette displays on the bottom left of the Pattern Builder. The topographical view of the virtual application pattern displays on the canvas.
4. Expand **Layers** to view the layers of the virtual application. Click the layer to view the topographical view on the canvas.

5. Click the **Create a new layer** icon. The new layer is added to the **Layers** palette.
   You can also add a layer by **importing an existing application** called a reference layer.

6. Click **Save**.

**Results**

You have created a layer for your virtual application.

**What to do next**

Edit the layer.

**Editing virtual application layers**

You can use the Pattern Builder in IBM Cloud Orchestrator to edit virtual application pattern layers.

**About this task**

By default, a virtual application includes one layer when you first create it. When you use application layering, you can modify an existing virtual application by adding separate layers. For example, you can use one virtual application that defines the basic components of one deployment environment to create a different virtual application for other deployment environments by associating the quality of service (QoS) layer with different QoS goals.

One virtual application can contain multiple layers. A layer can contain component types of the virtual application, or the layer can reference another application, which is then called a reference layer.

Because there is no predefined set of layers or binding between a component type and a particular layer, you can create layers according to your business goals. However, one component type in a virtual application can be placed in only one layer, but you can move parts between the layers.

In the Pattern Builder, you can add or delete a layer, edit a layer, disable or enable a layer, move virtual application parts between layers, or import a virtual application as a reference layer.

To edit a virtual pattern application layer:

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**. The Virtual Application Pattern palette displays.
2. Select the virtual application pattern for which you want to edit a layer.
3. Click the **Edit** icon. The Pattern Builder opens.
4. Expand **Layers** to view the layers of the virtual application pattern. Click the layer to view the topographical view on the canvas and to start editing.
5. Edit the layer. You can edit the layer in the following ways:
   - Rename the layer. Click one time on the name to modify.
   - Add or remove virtual application components.
Add or remove virtual application components connections.
Add or remove policies.
Move components between layers. Use the move to: icon to switch the layer group of each virtual application pattern part. When you select a virtual application pattern part and click the move to: icon, a list of layers displays. Select a layer to move the virtual application pattern part from the previous layer.

6. Click Save.

Results
You have edited an existing virtual application pattern layer.

What to do next
Deploy the virtual application pattern.

Deleting virtual application pattern layers
You can use the Pattern Builder in IBM Cloud Orchestrator to delete virtual application pattern layers.

About this task
In the Pattern Builder, you can add or delete a layer, edit a layer, disable or enable a layer, move virtual application parts between layers, or import a virtual application as a reference layer.

Procedure
1. Click PATTERNS > Pattern Design > Virtual Application Patterns. The Virtual Application Patterns palette displays.
2. Select the virtual application pattern for which you want to remove a layer.
3. Click the Edit icon. The Pattern Builder opens. The Layers palette displays on the lower left of the Pattern Builder. The topographical view of the virtual application pattern displays on the canvas.
4. Expand Layers to view the layers of the virtual application pattern.
5. Select the layer that you want to delete and click the Delete the selected layer icon. The topographical view displays on the canvas to show that the layer is deleted.

Results
You deleted a virtual application pattern layer.

Importing virtual application pattern layers
You can use the Pattern Builder in IBM Cloud Orchestrator to import an application as a reference layer.

About this task
In the Pattern Builder, you can add or delete a layer, edit a layer, disable or enable a layer, move virtual application patterns parts between layers, or import a virtual application pattern as a reference layer.
Use reference layers to reuse existing applications. Contents in the reference layer are read-only. Changes made from the referenced application are reflected in the application referencing it.

To import a new layer as a reference:

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**. The Virtual Application Patterns palette displays.
2. Select the virtual application pattern for which you want to import a layer.
3. Click the **Edit** icon. The Pattern Builder opens. The **Layers** palette displays on the lower left of the Virtual Application Patterns palette. The topographical view of the virtual application pattern displays on the canvas.
4. Expand **Layers** to view the layers of the virtual application pattern.
5. Click the **Import a virtual application** icon. The **Import Virtual Application** dialog box displays.
6. Select a virtual application that you want to reference as a layer and click **Add**. The new application layer is now listed under the **Layers** palette. If you select this layer, the topographical view displays on the canvas.

**Results**

You have imported a virtual application as a reference layer.

**What to do next**

After you import the reference layer, virtual application pattern components in other layers can connect to the reference layer.

**Working with virtual application templates**

The *virtual application template* is a predefined set of components and configuration used to simplify and standardize the creation of virtual application patterns.

**Before you begin**

You must be assigned the **catalogeditor** role or the **admin** role to work with virtual application templates.

**About this task**

When you design a virtual application template, you can configure it with default values or leave some values unconfigured. Application builders can use the virtual application template to create new virtual application patterns. virtual application templates also provide a more flexible deployment option. When you deploy from a virtual application template, you can specify property values that are not configured or edit values that are not locked.

To access the virtual application templates, click **PATTERNS > Pattern Design > Virtual Application Templates**.
Creating a virtual application template:

You can create a new virtual application template that is used to create a virtual application. The template can be saved in the IBM Cloud Orchestrator catalog.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

You can also use an existing virtual application template that was shipped with the product or you can create a virtual application template from an existing virtual application.

To create a virtual application template from an existing virtual application, click PATTERNS > Pattern Design > Virtual Application Patterns and select a virtual application in the list. Click the New icon to create a virtual application template. Click the Edit icon to start the Pattern Builder.

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Templates. The Virtual Application Templates palette displays.
2. Click the New icon on the toolbar.
3. To create your virtual application template:
   a. Select a pattern type from the drop-down menu.
   b. Select a virtual application template.
   c. Click Start Building. You have created a new virtual application template associated with a pattern type. The Pattern Builder opens in a new window where you can add components and policies.
4. On the Virtual Application properties pane, specify the following information:
   Name The name of the virtual application pattern.
   Description (Optional) The description of the virtual application pattern.
   Type Leave Application selected to create a virtual application pattern or select Template to create a virtual application template that is used as the basis for creating other virtual application patterns.
   Lock option for plugin usage
   Specify how this virtual application pattern is affected by upgrades to the pattern type or to IBM Foundation Pattern.
   Unlock plugins
   If the pattern type is upgraded use the latest versions of pattern type plug-ins. If IBM Foundation Pattern is upgraded, use the latest version.
   Lock all plugins
   Do not change the version of plug-ins or the version of the IBM Foundation Pattern associated with this virtual application pattern when an upgrade occurs.
   Lock all plugins except Foundation plugins
   If the pattern type is upgraded, do not change the version of the plug-ins associated with this virtual application pattern. If IBM Foundation Pattern is upgraded, use the latest version.
Note: You can view a list of which plug-ins are locked if you select Lock all plugins or Lock all plugins except Foundation plugins. Click the Source tab in Pattern Builder. The application model source is displayed. Search for the element plugins to view the list.

5. Design your virtual application template with the components, links, and policies that you want to include.

6. Click Save.

What to do next

The new application template is now available in the catalog and can be used to create a virtual application pattern.

Creating virtual applications from templates:

You can use an existing virtual application template to create a virtual application. These templates are either templates you have already created or is a template that was shipped with the product.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

About this task

You can deploy a virtual application directly from a template or use it as a starting point to build a virtual application pattern.

If you do not want to use an existing virtual application template, you can create a new virtual application template.

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Templates. The Virtual Application Templates palette displays.
2. Select the pattern type associated with the template or search for the template in the left pane.
3. Select an existing template.
4. Click Open on the toolbar to edit the template.
5. Edit the virtual application pattern component parts:
   a. Click the virtual application pattern listed in the left navigation.
   b. Click the Open icon in the Virtual Application Pattern pane. The Pattern Builder displays with the list of virtual application components that you can drag to the canvas to customize your virtual application pattern.
   c. Select a virtual application component and drag the component onto the canvas to build your virtual application.
   d. To add a policy, click Add policy for application icon.
   e. To create the connection between the parts, hover over one of the objects until the blue circle turns orange. Select the circle with the left mouse button, drag a connection to the second object until the object is highlighted, and release the mouse button.
6. Click Save.
Cloning a virtual application template:

You can clone a virtual application template to create a copy that you can edit and customize. The template is added to the catalog.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Templates. The Virtual Application Templates palette displays.
2. Select a virtual application template and click the Clone icon on the toolbar.
3. Click OK to confirm that you want to clone the virtual application template.
4. Click Open to name and edit the template.
5. On the Virtual Application properties pane, specify the following information:
   - **Name**  The name of the virtual application pattern.
   - **Description** (Optional) The description of the virtual application pattern.
   - **Type** Leave Application selected to create a virtual application pattern or select Template to create a virtual application template that is used as the basis for creating other virtual application patterns.

   **Lock option for plugin usage**
   Specify how this virtual application pattern is affected by upgrades to the pattern type or to IBM Foundation Pattern.
   - **Unlock plugins** If the pattern type is upgraded use the latest versions of pattern type plug-ins. If IBM Foundation Pattern is upgraded, use the latest version.
   - **Lock all plugins** Do not change the version of plug-ins or the version of the IBM Foundation Pattern associated with this virtual application pattern when an upgrade occurs.
   - **Lock all plugins except Foundation plugins** If the pattern type is upgraded, do not change the version of the plug-ins associated with this virtual application pattern. If IBM Foundation Pattern is upgraded, use the latest version.

   **Note:** You can view a list of which plug-ins are locked if you select Lock all plugins or Lock all plugins except Foundation plugins. Click the Source tab in Pattern Builder. The application model source is displayed. Search for the element plugins to view the list.
6. Click Save.
Editing virtual application templates:

You can edit a virtual application template and use it to create a virtual application pattern. The template can be saved in the catalog. The virtual application template is necessary to build your virtual application pattern that is eventually deployed as a virtual application instance.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Templates. The Virtual Application Templates palette displays.
2. Select a virtual application template. The template details display to the right.
3. Click the Open icon on the toolbar to edit the template. The Pattern Builder opens.

Importing a virtual application template:

You can import a virtual application template to the IBM Cloud Orchestrator catalog. The virtual application template is necessary to build your virtual application pattern that is eventually deployed as a virtual application instance.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

About this task

To import a virtual application template with the user interface:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Templates. The Virtual Application Templates palette displays.
2. Click the Import icon. The Import Application dialogue window opens.
3. Select an application compressed file to import. Click Browse to find the application file. Your system proceeds to upload the compressed application file.

Results

You have imported an application template.
Exporting a virtual application template:

You can export a virtual application template from the IBM Cloud Orchestrator catalog. The virtual application template is necessary to build your virtual application pattern that is eventually deployed as a virtual application instance.

Before you begin

You must be assigned the catalogeditor role or the admin role to perform these steps.

About this task

To export a virtual application template with the user interface:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Templates. The Virtual Application Templates palette displays.
2. Select an existing template from the left panel of the Virtual Application Templates palette.
3. Click the Export icon.

Results

You have exported a virtual application template.

Removing a virtual application template:

You can remove a virtual application template from the IBM Cloud Orchestrator catalog when it is no longer needed.

Before you begin

You must be assigned the catalogeditor role and be granted all access to the application template to remove. You can also perform these steps if you are assigned the admin role.

About this task

To remove a virtual application template with the user interface:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Templates. The Virtual Application Templates palette displays.
2. From the Virtual Application Templates palette, select a virtual_application_template to remove from the catalog.
3. Click the Delete icon to remove the virtual application template from the catalog. A window displays requesting your confirmation that this application template can permanently be removed.
4. Click OK to confirm that you want to remove the virtual application template.

Results

The virtual application template has been removed from the catalog.
Working with virtual application pattern plug-ins

Plug-ins provide the constituent parts of a virtual application, as well as the underlying implementation that makes the application deployable in the cloud. Pattern types, the containers of solution-specific and topology-specific resources that are required for different types of virtual applications, are collections of plug-ins.

About this task

Plug-ins contribute components, links and policies that appear in the Pattern Builder. They are grouped into pattern types. When a virtual application builder creates a virtual application pattern, the first step is choosing the pattern type. This choice determines the options and the user experience in the Pattern Builder. In the Pattern Builder, virtual application builders can select from the components, links and policies that the plug-ins in the pattern type expose. The plug-in that contributes a component or link completely determines its semantics and operation. Components, links and policies are the most user-visible capabilities a plug-in can contribute, but there are other capabilities that a plug-in developer must include. Plug-ins are responsible for the implementation of components and links when a virtual application is deployed, and maintenance through the entire lifecycle of the virtual application. A plug-in must contribute proper lifecycle scripts to manage the virtual application through its various lifecycle events.

A pattern type is a collection of plug-ins that are designed for a specific type of virtual application pattern and used as the foundation of a virtual application. For example, a web application uses the IBM Web Application Pattern (not released with the product), and a database uses the IBM Database Application Patterns (not released with the product). When a user selects a pattern type in the Pattern Builder, the design experience is determined by the associated plug-ins. Pattern types are purchasable at the IBM PureSystems Centre.

You can work with plug-ins as follows:

- Use an existing plug-in to create or extend a virtual application
- Developing plug-ins to create or extend a virtual application

Managing system plug-ins

IBM Cloud Orchestrator includes a set of preinstalled system plug-ins. These plug-ins contain the necessary code for component parts when building virtual application patterns. The plug-in controls the end-to-end processing and implementation of the component parts that you use to build the virtual application pattern. Plug-ins also contribute components, links, and policies that you can choose to customize your virtual application pattern.

Before you begin

To manage a plug-in, you must be assigned the admin role.

Important: A plug-in is disabled if any of its attributes (configuration parameters) are not specified.

About this task

You can use either the user interface, the command-line interface, or the REST API to manage your plug-ins.
To learn more about using the command-line interface see command-line interface.

After setting up system plug-ins, you can build a virtual application pattern with component parts or edit an existing virtual application pattern. After enablement by the administrator, you can select the specified version and list all the plug-ins with the IBM version level, release, modification and fix level (v.r.m.f) structure format.

**Plug-ins shipped with pattern types:**

Several preinstalled system plug-ins are available with the Foundation pattern type shipped with IBM Cloud Orchestrator pattern types. You can use these plug-ins to extend the function of virtual applications.

In the IBM Cloud Orchestrator user interface, click **PATTERNS > Deployer Configuration > System Plug-ins** and select the Foundation pattern type to see a list of the related plug-ins.

In the Pattern Builder, components are grouped into categories.

**Adding system plug-ins to the catalog:**

You can add a plug-in to the catalog. Plug-ins define components, links, and policies for virtual application patterns.

**Before you begin**

You must be assigned the catalogeditor role or the admin role to perform these steps.

**Attention:** Plug-ins are disabled when configurations are not completed.

**Procedure**
1. Open the IBM Cloud Orchestrator user interface.
2. Click **PATTERNS > Deployer Configuration > System Plug-ins**.
3. Select the Add icon to upload the plug-in .tgz file. A dialogue box displays where you can browse for a plug-in .tgz file to import.

   **Important:** If the .tgz file is more than 2 GB, use the command-line interface to upload the plug-in file to your system.

**Deleting plug-ins from the catalog:**

You can remove a plug-in from the catalog when it is no longer needed.

**Before you begin**

You must be assigned the catalogeditor role or the admin role to perform these steps.

**Procedure**
1. Open the IBM Cloud Orchestrator user interface.
2. Click **PATTERNS > Deployer Configuration > System Plug-ins**.
3. Select the Delete icon to delete the plug-in file. A window displays requesting confirmation that you want to remove the plug-in.
4. Click **OK** to confirm that you want to remove the plug-in.

**Virtual application pattern components**

A virtual application pattern contains components that represent middleware services that are required by the virtual appliance instance.

You can connect components in a virtual application pattern to indicate dependencies and optionally apply a *policy* to configure middleware services during deployment to configure specific behavior or define a quality of service level. Components, links, and policies can have required and optional attributes.

Components, links, and policies are defined by *plug-ins*. When you create a virtual application pattern, the available components, links, policies, and configuration options are determined by the plug-ins that are included with the selected pattern type.

**Components**

The following components are available with the virtual application patterns provided with IBM Cloud Orchestrator or purchasable at the IBM PureSystems Centre.

**Important:** Components that require disk add-ons are not supported in IBM Cloud Orchestrator.

- Application
  - Additional archive file
  - Enterprise application
  - “Existing web service provider endpoint” on page 501
  - “Policy set” on page 502
  - Web application such as IBM WebSphere Application Server
- Database
  - Data Studio web console
  - Database such as IBM DB2
  - “Existing database (DB2)” on page 512
  - Existing database (Informix)
  - Existing database (Oracle)
  - “Existing IMS database” on page 517
- Messaging
  - “Existing Messaging Service (WebSphere MQ)” on page 529
  - “Existing queue (WebSphere MQ)” on page 532
  - “Existing topic (WebSphere MQ)” on page 531
- OSGi
  - External OSGi bundle repository
  - OSGi application
- Transaction Processing
  - CICS Transaction Gateway
  - “Existing IMS TM” on page 541
- User Registry
  - “Existing User Registry (IBM Tivoli Directory Server)” on page 518
  - “Existing User Registry (Microsoft Active Directory)” on page 522
Policies

You can optionally apply policies to a virtual application to configure specific behavior in the deployed virtual application instance. Two virtual applications might include identical components, but require different policies to achieve different service level agreements. For example, if you want a web application to be highly available, you can add a scaling policy to the web application component and specify requirements such as a processor usage threshold to trigger scaling of the web application. At deployment time, the topology of the virtual application is configured to dynamically scale the web application. Multiple WebSphere Application Server instances are deployed initially for the web application and instances are added and removed automatically based on the service levels that are defined in the policy.

Policies can be applied only to particular types of components. For more information, see the following links:

- Scaling policy
- Routing policy
- Java virtual machine (JVM) policy
- Log policy

Application components:

There are several application components to choose from when building a virtual application pattern.

About this task

- Additional archive file
- Enterprise application component
- Existing web service provider endpoint
- Policy set
- Web application component

Additional archive file:

The additional archive file component is for your primary archive.

Before you begin

The following is a required attribute for a additional archive file:

- **Additional archive file**: Specifies the external archive file that contains additional files needed by the web archive (WAR) or enterprise archive (EAR) file.

- **Extraction path**: Specifies the extraction path for the additional archive file. This path must not be an existing path.
Table 42. Incoming connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web archive (WAR files).</td>
<td>• Provider policy set binding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Service name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Binding file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Key store</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Trust store (encryption)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Trust store (digital signature)</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise archive (EAR files).</td>
<td>• Provider policy set binding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Service name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Binding file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Key store</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Trust store (encryption)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Trust store (digital signature)</td>
</tr>
</tbody>
</table>

About this task

You can view, edit or add this virtual application component in the user interface as follows:

Procedure

1. Click PATTERNs > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing archive file component, select the Additional archive file component part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new archive file component to a virtual application pattern, click the Additional archive file component listed under the Application Component and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the additional archive file component properties by viewing the plug-in information. Click PATTERNs > Deployer Configuration > System Plug-ins. Select file/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.
Enterprise application component:

The enterprise application (WebSphere Application Server) component represents an execution service for Java Platform, Enterprise Edition (Java EE) enterprise application (EAR) files.

Before you begin

Attention: You cannot use an enterprise application that includes Container Managed Persistence V 2.0 beans. This type of application requires deploy tools that are not included in this product’s WebSphere Application Server binary files.

The following are attributes for an enterprise application:

- **EAR file**: Specifies the enterprise archive (.ear) file to be uploaded. This attribute is required.
- **Total transaction lifetime timeout**: Specifies the default maximum time, in seconds, allowed for a transaction that is started on this server before the transaction service initiates timeout completion. Any transaction that does not begin completion processing before this timeout occurs is rolled back. The default is 120 seconds.
- **Asynchronous response timeout**: Specifies the amount of time, in seconds, that the server waits for responses to WS-AT protocol messages. The default is 120 seconds.
- **Client inactivity timeout**: Specifies the maximum duration, in seconds, between transactional requests from a remote client. Any period of client inactivity that exceeds this timeout results in the transaction being rolled back in this application server. The default is 60 seconds.
- **Maximum transaction timeout**: Specifies, in seconds, the maximum transaction timeout for transactions that run in this server. This value is greater than, or equal to, the value specified for the total transaction timeout. The default is 300 seconds.
- **Iterim fixes URL**: Specifies the location or URL of the selected interim fixes. This URL is used by the WebSphere Application Server virtual machine to download interim fixes for update.

Policies

Table 43. Policy components for enterprise applications

<table>
<thead>
<tr>
<th>Policy name</th>
<th>Description</th>
</tr>
</thead>
</table>
### Table 44. Incoming connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application, such as a WebSphere Application Server application, cloud component represents an execution service for Java EE enterprise applications (EAR) files.</td>
</tr>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java EE web applications (WAR) files.</td>
</tr>
<tr>
<td>System updates</td>
<td>System updates for download and update on the virtual machine.</td>
</tr>
</tbody>
</table>

### Table 45. Outgoing connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing Topic (WebSphere MQ)</td>
<td>An existing topic represents a message destination on an external IBM WebSphere MQ messaging service through which messages are published and subscribed.</td>
</tr>
<tr>
<td>Additional archive file</td>
<td>An additional archive file component for your primary archive.</td>
</tr>
<tr>
<td>Existing messaging service (WebSphere MQ)</td>
<td>A messaging service represents a connection to an external messaging system such as WebSphere MQ.</td>
</tr>
<tr>
<td>Policy set</td>
<td>A component used to define quality of service policies.</td>
</tr>
<tr>
<td>Existing database (Oracle)</td>
<td>An existing Oracle database component represents a connection to an existing Oracle database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote Oracle database.</td>
</tr>
<tr>
<td>Generic target</td>
<td>A component used to open the firewall for outbound TCP connections from a web or enterprise application to a specified host and port.</td>
</tr>
<tr>
<td>Component name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Database (DB2)                | A database (DB2) component that represents a pattern-deployed database service.                                                                                                                         | • JNDI name of the data source  
• Resource references of the data source  
• Non-transactional data source  
• Minimum connections  
• Maximum connections  
• Connection timeout                                                                                     |
| Existing database (DB2)       | An existing DB2 database component represents a connection to a remote DB2 database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote DB2 database. |                                                                                                       |
| Existing database (Informix®)  | An existing Informix database component represents a connection to a remote Informix database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote Informix database. |                                                                                                       |
| Existing CICS Transaction Gateway (CTG) | An existing CTG component represents a connection to an existing CTG instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the CTG. |                                                                                                       |
| Existing IMS™ database         | Existing Information Management Systems (IMS) database system.                                                                                                                                              |                                                                                                       |
| Existing user registry (IBM Tivoli Directory Server) | An existing user registry, such as Lightweight Directory Access Protocol (LDAP), cloud component represents a pattern-deployed LDAP service that can be deployed by itself or attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security. | • User filter  
• Group filter  
• Role name  
• User role mapping  
• Group role mapping  
• Special subject mapping                                                                                   |
To make a connection between a component and the enterprise application, hover over the blue circle on the enterprise application component part on the canvas. When the blue circle turns yellow, draw a connection between the enterprise application and component.

Use the property panel to upload the EAR files. To make associations with other services, create a link to the corresponding virtual application pattern component.
At this time, support is limited to one database and one user registry connection. A scaling policy object might be attached to specify a highly available pattern.

**CAUTION:** When using an enterprise application to deploy a database component, you define the database schema in the SQL file. Do not add the “connect to <dbname>” statement in the SQL file. The schema object that is used is db2inst1 and not appuser.

In addition to uploading your EAR file, you can upload additional files, such as a compressed file containing configuration details or other information. When the WebSphere process starts, the compressed file is extracted to a directory and the `icmp.external.directory` system property is set. If you attach a scaling policy to the web application component, each virtual machine contains a copy of the compressed file, and any updates made to the file or directory on one virtual machine are not reflected in the copy of the file on another virtual machine.

By default, the application is available at http://{ip_address}/{context_root}, where:
- `{ip_address}` is obtained from the list of deployed cloud applications
- `{context_root}` is specified within the EAR file.

**About this task**

You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**
1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing enterprise application component, select the Enterprise Application component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a enterprise application component to a virtual application pattern, click the Enterprise Application component listed under the Application Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the enterprise application component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select `was/h.x.x.x` from the System Plug-ins palette where `x.x.x.x` corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.
**Existing web service provider endpoint:**

A web service provider endpoint is a web service provider that is provided by remote server.

**Before you begin**

**Important:** To use the Web service plug-in in IBM Cloud Orchestrator, the web service client must be updated as follows:

1. Include the Web service WSDL file in the application package. The WSDL file is located in `WEB-INF/wsdl` directory.
2. Update the Web service client source code to specify the location of the WSDL file. For example: "file:/WEB-INF/wsdl/{WSDL_FILE_NAME}"

The following is a required attribute for a Web service provider endpoint component:

- **Host (IP):** Specifies the Host IP of the remote server.
- **Port:** Specifies the port of remote server.

**Connections**

*Table 46. Incoming connectable components*

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) enterprise archive (EAR files).</td>
<td>• Service name</td>
</tr>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application (WebSphere Application Server) cloud component represents an execution service for Java EE web archive (WAR files).</td>
<td>• Service name</td>
</tr>
</tbody>
</table>

To make a connection between a component and the enterprise application, hover over the blue circle on the enterprise application component part on the canvas. When the blue circle turns yellow, draw a connection between the enterprise application and component.

**About this task**

You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**.
2. Select a **virtual_application_pattern**.
3. Click **Edit the virtual application** icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing enterprise application component, select the **Existing Web Service Provider Endpoint** component part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.

5. To add a the existing Web service provider endpoint component to a virtual application pattern, click the **Existing Web Service Provider Endpoint** component listed under the **Application Components** and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.

6. You can also view the enterprise application component properties by viewing the plug-in information. Click **PATTERNS > Deployer Configuration > System Plug-ins**. Select `webservice/x.x.x.x` from the **System Plug-ins** palette where `x.x.x.x` corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.

**Policy set:**

A policy set is a component that is used to define quality of service (QoS) policies. It is a collection of assertions about how services are defined, which can be used to simplify security configurations.

**Before you begin**

For more details about policy sets in IBM WebSphere Application Server, see the [WebSphere Application Server Information Center](https://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.wps.doc/ae/ae-trg-appserv.html). Refer to the topics, Managing policy sets using the administrative console, and Exporting policy sets using the administrative console. The required policy set zip files can be retrieved export topic.

The following is a required attribute for a policy set component:

* **Policy Set File**: Specifies the policy set file to upload.

**Connections**

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) enterprise archive (EAR files). | • Provider policy set binding  
• Service name  
• Binding file  
• Key store  
• Trust store (encryption)  
• Trust store (digital signature) |
Table 47. Incoming connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application (WebSphere Application Server) cloud component represents an execution service for Java EE web archive (WAR files).</td>
<td>• Provider policy set binding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Service name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Binding file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Key store</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Trust store (encryption)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Trust store (digital signature)</td>
</tr>
</tbody>
</table>

To make a connection between a component and the policy set, hover over the blue circle on the enterprise application component part on the canvas. When the blue circle turns yellow, draw a connection between the policy set and component.

About this task

You can view, edit, or add this virtual application component in the user interface as follows:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing policy set component, select the Policy Set component part on the Pattern Builder canvas. The properties panel displays.
   
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new policy set component to a virtual application pattern, click the Policy Set component listed under the Application Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the policy set component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select webservice/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.
Web application component:

The web application component represents an execution service for the Java Platform, Enterprise Edition (Java EE) web archive (WAR) files.

Before you begin

The following is a required attribute for a web application:
- **WAR file**: Specifies the web archive (WAR) file to be uploaded. This attribute is required.
- **Context root**: Specifies the context root of the web module. The attribute applies to the WAR file only.
- **Interim fixes URL**: Specifies the location of the URL of selected interim fixes. This URL is used by the WebSphere Application Server virtual machine to download interim fixes.

Policies

*Table 48. Policy components for web applications*

<table>
<thead>
<tr>
<th>Policy name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing policy (web, enterprise, or OSGi enterprise bundle archive (EBA) application)</td>
<td>Routing policy for a web application, enterprise application, or an OSGi EBA application.</td>
</tr>
<tr>
<td>Log policy (web or enterprise application)</td>
<td>A policy to specify configuration for log files.</td>
</tr>
<tr>
<td>JVM policy (web or enterprise application)</td>
<td>A policy to control features of the underlying Java Virtual Machine (JVM).</td>
</tr>
<tr>
<td>Scaling policy (web or enterprise application)</td>
<td>Scaling is a run time capability to automatically scale your application platform as the load changes. A scaling policy component defines this capability and the conditions under which scaling activities are performed for your application.</td>
</tr>
</tbody>
</table>

Connections

*Table 49. Incoming connectable components*

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application, such as a WebSphere Application Server application, cloud component represents an execution service for Java EE enterprise applications (EAR) files.</td>
</tr>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java EE web applications (WAR) files.</td>
</tr>
<tr>
<td>System updates</td>
<td>System updates for download and update on the virtual machine.</td>
</tr>
</tbody>
</table>
Table 50. Outgoing connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing Topic (WebSphere MQ)</td>
<td>An existing topic represents a message destination on an external IBM WebSphere MQ messaging service through which messages are published and subscribed.</td>
<td>• Java Naming and Directory Interface (JNDI) name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Resource environment references</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Message destination references</td>
</tr>
<tr>
<td>Additional archive file</td>
<td>An additional archive file component for your primary archive.</td>
<td></td>
</tr>
<tr>
<td>Existing messaging service (WebSphere MQ)</td>
<td>A messaging service represents a connection to an external messaging system such as WebSphere MQ.</td>
<td>• JNDI name of the Java Message Service (JMS) connection factory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Resource references of the JMS connection factory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Client ID</td>
</tr>
<tr>
<td>Policy set</td>
<td>A component used to define quality of service policies.</td>
<td></td>
</tr>
<tr>
<td>Existing database (Oracle)</td>
<td>An existing Oracle database component represents a connection to an existing Oracle database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote Oracle database.</td>
<td></td>
</tr>
<tr>
<td>Generic target</td>
<td>A component used to open the firewall for outbound TCP connections from a web or enterprise application to a specified host and port.</td>
<td></td>
</tr>
<tr>
<td>Database (DB2)</td>
<td>A database (DB2) component that represents a pattern-deployed database service.</td>
<td>• JNDI name of the data source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Resource references of the data source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Non-transactional data source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minimum connections</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Maximum connections</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Connection timeout</td>
</tr>
<tr>
<td>Existing database (DB2)</td>
<td>An existing DB2 database component represents a connection to a remote DB2 database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote DB2 database.</td>
<td></td>
</tr>
<tr>
<td>Component name</td>
<td>Description</td>
<td>Connection</td>
</tr>
<tr>
<td>-------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Existing database (Informix)</td>
<td>An existing Informix database component represents a connection to a remote Informix database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote Informix database.</td>
<td></td>
</tr>
<tr>
<td>Existing CICS Transaction Gateway (CTG)</td>
<td>An existing CTG component represents a connection to an existing CTG instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the CTG.</td>
<td></td>
</tr>
<tr>
<td>Existing Information Management Systems (IMS) database</td>
<td>Existing IMS database system.</td>
<td></td>
</tr>
<tr>
<td>Existing user registry (Tivoli Directory Server)</td>
<td>An existing user registry, such as Lightweight Directory Access Protocol (LDAP), cloud component represents a pattern-deployed LDAP service that can be deployed by itself or attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security.</td>
<td>• User filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Group filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Role name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• User role mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Group role mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Special subject mapping</td>
</tr>
<tr>
<td>Existing user registry (Microsoft Active Directory)</td>
<td>An existing user registry, such as LDAP, represents an existing LDAP service that can be attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container managed security.</td>
<td></td>
</tr>
</tbody>
</table>
Table 50. Outgoing connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>User registry (Tivoli Directory Server)</td>
<td>A user registry, such as Tivoli Directory Server, cloud component represents a pattern-deployed LDAP service that can be deployed by itself or attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security.</td>
<td></td>
</tr>
<tr>
<td>Existing IMS Transaction Manager</td>
<td>Existing IMS TM</td>
<td></td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application</td>
<td>An enterprise application, such as a WebSphere Application Server application, cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
<td></td>
</tr>
<tr>
<td>Server)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application, such as a WebSphere Application Server application, cloud component represents an execution service for Java EE web applications (WAR files).</td>
<td></td>
</tr>
<tr>
<td>Existing web service provider endpoint</td>
<td>A web service provider provided by a remote server.</td>
<td></td>
</tr>
</tbody>
</table>
| Existing queue (WebSphere MQ)                 | A message queue on an external WebSphere MQ messaging service through which messages are sent and received.                                                                                               | • JNDI name  
• Resource environment references  
• Message destination references |

Use the property panel to upload the WAR files. You can also specify a context root. To make associations with other services, create a link to the corresponding cloud component. At this time, support is limited to one database and one user registry connection. A high availability (HA) policy object might be attached to specify an HA pattern.

In addition to uploading your WAR file, you can upload additional files, such as a compressed file containing configuration details or other information. When the WebSphere process starts, the compressed file is extracted to a directory and the `icmp.external.directory` system property is set. If you attach an HA policy to the web application component, each virtual machine contains a copy of the compressed file, and any updates made to the file or directory on one virtual machine are not reflected in the copy of the file on another virtual machine.

By default, the application is available at `http://{ip_address}/{context_root}`, where:

- `{ip_address}` is obtained from the list of deployed virtual application patterns.
{context_root} is user-specified for WAR files or specified within the enterprise archive (EAR) file.

**About this task**

You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a *virtual_application_pattern*.
3. Click *Edit the virtual application* icon located in the top right corner of the Pattern Builder palette.
4. To edit an existing web application component, select the **Web Application** component part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a web application component to a virtual application pattern, click the **Web Application** component listed under the **Application Components** and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the web application component properties by viewing the plug-in information. Click PATTERNS > **Deployer Configuration** > **System Plug-ins**. Select **was/x.x.x.x** from the **System Plug-ins** palette where *x.x.x.x* corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.

**Database components:**

There are several database components to choose from when building a virtual application pattern.

**About this task**

- "Database (DB2)" on page 510
- "Existing database (DB2)" on page 512
- "Existing database (Oracle)" on page 515
- "Existing IMS database" on page 517
- "Existing database (Informix)" on page 513
- "Database Studio web console" on page 509
**Database Studio web console:**

The Database Studio web console component is a database tool included with the IBM Database Patterns.

**Before you begin**

The IBM Database Patterns are purchasable at the [IBM PureSystems Centre](#). This plug-in component is not available on the Pattern Builder unless you have accepted the license for the IBM Database Patterns.

The following are the attributes for a component:

- **Data Studio web console administrator user**: Specifies the Data Studio web console administrator user ID.
- **Data Studio web console administrator user password**: Specifies Data Studio web console administrator user password.

To make a connection between an application component and the database, hover over the blue circle on the Data Studio web console component part on the canvas. When the blue circle turns yellow, draw a connection between the Data Studio web console and the application component.

**About this task**

You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**.
2. Select a **virtual_application_pattern**.
3. Click **Edit the virtual application** icon located in the top right corner of the Pattern Builder palette.
4. To edit an existing Data Studio web console component, select the **Data Studio web console** component part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a Data Studio web console database component to a virtual application pattern, click the **Data Studio web console** component listed under the **Database Components** and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the Data Studio web console component properties by viewing the plug-in information. Click **PATTERNS > Deployer Configuration > System Plug-ins**. Select `dswc/x.x.x.x` from the System Plug-ins palette where `x.x.x.x` corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.
Database (DB2):

The DB2 database component represents a pattern-deployed database service.

Before you begin

The following are the attributes for a DB2 database component:

- **Database name**: Specifies the name of the database name that you want to deploy.
- **Database description**: Specifies a description of the database that you want to deploy.
- **Purpose**: Specifies the purpose for the database. Select *Production* or *Non-Production*. The default value is *Production*.
- **Source**: Select one of the following sources:
  - Clone from a database image: Specifies a clone from the database image.
  - Apply database standard: Specifies to apply a database standard.

Settings include:

- **Maximum User Data Space (GB)**: Specifies the maximum size of the data space, in gigabytes, in the database that you want to deploy. The default value is 10 GB.

- **Workload standards**: Specifies the workload standards. Settings include:
  - departmental OLTP: Specifies the departmental OLTP standard. The workload type is Departmental OLTP.
  - dynamic_datamart: Specifies the dynamic data mart standard. The workload type Dynamic data mart.

- **DB2 Compatability Mode**: Specifies the DB2 compatibility mode. Select *Default Mode* or *Oracle Mode*. The default value is *Default Mode*.

- **Schema file**: Specifies the schema file (*.ddl, *.sql) that defines the database schema. Click **Browse** to search for the file on your system.

Connections

*Table 51. Incoming connectable components*

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files).</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
</tr>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server.</td>
</tr>
</tbody>
</table>
To make a connection between an application component and the database, hover
over the blue circle on the DB2 database component part on the canvas. When the
blue circle turns yellow, draw a connection between the DB2 database and the
application component.

The application is assumed to use Java Naming and Directory Interface (JNDI)
settings to locate the data source. Specify the JNDI name in the link property
panel. During deployment, the JNDI name is set to the corresponding data source,
and the name must match the name that is coded into the application.

About this task

You can view, edit, or add this virtual application component in the user interface
as follows:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the
   Pattern Builder palette.
4. To edit an existing DB2 database component, select the Database (DB2)
   component part on the Pattern Builder canvas. The properties panel is
displayed. For more details on the properties panel settings, view the help by
selecting the help icon on the properties panel.
5. To add a DB2 database component to a virtual application pattern, click the
   Database (DB2) component listed under the Database Components and drag
   the icon to the Pattern Builder canvas. The properties panel for the component
   is displayed to the right of the Pattern Builder palette. For more details on the
   properties panel settings, view the help by selecting the help icon on the
   properties panel.

   **CAUTION:** When defining the database schema, do not add the "connect to
   <dbname>" statement in the SQL file. The schema object that is used is
db2inst1 and not appuser.
6. You can also view the DB2 database component properties by viewing the
   plug-in information. Click PATTERNS > Deployer Configuration > System
   Plug-ins. Select db2/x.x.x.x from the System Plug-ins palette where x.x.x.x
   corresponds to the version numbers. The component plug-in configuration
   information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

View the agent log files for troubleshooting. To find the appropriate trace.log file,
navigate to /logs/database-<db_name>.<instanceNumber>.<DB2>.
**Existing database (DB2):**

An existing DB2 database component represents a connection to a remote DB2 database instance running remotely outside of the cloud infrastructure. The configuration properties allow a connection to be made to the remote DB2 database.

**Before you begin**

The following are the attributes for a remote DB2 database component:

- **Existing database name:** Specifies the name of the existing DB2 database. This attribute is required.
- **Server host name or IP:** Specifies the server host name or IP address of the existing DB2 database. This attribute is required.
- **Server port number:** Specifies the port number of the existing DB2 database. The default is 50000. This attribute is required.
- **User name:** Specifies the user name to access the existing DB2 database. This attribute is required.
- **Password:** Specifies the password to access the existing DB2 database. This attribute is required.

**Connections**

*Table 52. Incoming connectable components*

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Web application (WebSphere Application Server) | A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files). | • Java Naming and Directory Interface (JNDI) name of the data source  
• Resource references of the data source  
• Non-transactional data source |
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files). | • JNDI name of the data source  
• Resource references of the data source  
• Non-transactional data source |
| OSGi application (WebSphere Application Server) | OSGi application on WebSphere Application Server. | • JNDI name of the data source  
• Resource references of the data source  
• Non-transactional data source |

To make a connection between an application component and the remote database, hover over the blue circle on the DB2 remote database component part on the canvas. When the blue circle turns yellow, draw a connection between the DB2 remote database and the application component.

The application is assumed to use JNDI settings to locate the data source. Specify the JNDI name in the link property panel. During deployment, the JNDI name is
set to the corresponding data source, and the name must match the name that is coded into the application.

**About this task**

You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**.
2. Select a *virtual_application_pattern*.
3. Click **Edit the virtual application** icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing remote DB2 database component, select the **Existing Database** component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add an existing DB2 database component to a virtual application pattern, click the **Existing Database (DB2)** component listed under the **Database Components** and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the remote DB2 database component properties by viewing the plug-in information. Click **PATTERNS > Deploier Configuration > System Plug-ins**. Select `wasdb2/x.x.x` from the **System Plug-ins** palette where `x.x.x` corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.

**Existing database (Informix):**

An existing Informix database component represents a connection to a remote Informix database running remotely outside of the cloud infrastructure. The configuration properties allow a connection to be made to the remote Informix database.

**Before you begin**

The following are the attributes for a remote Informix database component:

- **Database name**: Specifies the name of the existing Informix database. This attribute is required.
- **Server host name or IP address**: Specifies the server host name or IP address of the existing Informix database. This attribute is required.
- **Server port number**: Specifies the port number of the existing Informix database. The default is 9088. This attribute is required.
- **User name**: Specifies the user name to access the existing Informix database. This attribute is required.
• **Password:** Specifies the password to access the existing Informix database. This attribute is required.

### Connections

**Table 53. Incoming connectable components**

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Web application (WebSphere Application Server)      | A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files). | • Java Naming and Directory Interface (JNDI) name of the data source  
• Resource references of the data source  
• Non-transactional data source |
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files). | • JNDI name of the data source  
• Resource references of the data source  
• Non-transactional data source |
| OSGi application (WebSphere Application Server)      | OSGi application on WebSphere Application Server.                           | • JNDI name of the data source  
• Resource references of the data source  
• Non-transactional data source |

To make a connection between an application component and the remote database, hover over the blue circle on the Informix remote database component part on the canvas. When the blue circle turns yellow, draw a connection between the Informix remote database and the application component.

The application is assumed to use JNDI settings to locate the data source. Specify the JNDI name in the link property panel. During deployment, the JNDI name is set to the corresponding data source, and the name must match the name that is coded into the application.

### About this task

You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**.
2. Select a **virtual_application_pattern**.
3. Click **Edit the virtual application** icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing remote Informix database component, select the **Existing Database** component part on the Pattern Builder canvas. The properties panel displays.

   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add an existing Informix database component to a virtual application pattern, click the **Existing Database (Informix)** component listed under the **Database Components** and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.

6. You can also view the remote Informix database component properties by viewing the plug-in information. Click **PATTERNS > Deployer Configuration > System Plug-ins**. Select `wasdb2/x.x.x.x` from the **System Plug-ins** palette where `x.x.x.x` corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.

**Existing database (Oracle):**

An existing Oracle database component represents a connection to an existing Oracle database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote Oracle database.

**Before you begin**

The following are the attributes for an existing Oracle database component:

- **Database name**: Specifies the name of the existing Oracle database.
- **Server host name or IP address**: Specifies the server host name or IP address of the existing Oracle database server. This attribute is required.
- **Server port number**: Specifies the port number of the existing Oracle database. The default is 1521. This attribute is required.
- **User name**: Specifies the name used to access the existing Oracle database. This attribute is required.
- **Password**: Specifies the password associated with the existing Oracle database. This attribute is required.

**Connections**

**Table 54. Incoming connectable components**

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Web application (WebSphere Application Server) | A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web archive (WAR files). | • Java Naming and Directory Interface (JNDI) name of the data source  
  • Resource references of the data source  
  • Non-transactional data source |

| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise archive (EAR files). | • JNDI name of the data source  
  • Resource references of the data source  
  • Non-transactional data source |
Table 54. Incoming connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server.</td>
<td>• JNDI name of the data source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Resource references of the data source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Non-transactional data source</td>
</tr>
</tbody>
</table>

To make a connection between an application component and the existing Oracle database, hover over the blue circle on the existing Oracle database component part on the canvas. When the blue circle turns yellow, draw a connection between the existing Oracle database and the application component.

The application is assumed to use JNDI settings to locate the data source. Specify the JNDI name in the link property panel. During deployment, the JNDI name is set to the corresponding data source, and the name must match the name that is coded into the application.

About this task

You can view, edit, or add this virtual application component in the user interface as follows:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing remote Oracle database component, select the Existing Database (Oracle) component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add an existing Oracle database component to a virtual application pattern, click the Existing Database (Oracle) component listed under the Database Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on these the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the Oracle database component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select wasoracle:x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.
**Existing IMS database:**

An Information Management Systems Database IMS DB component represents a connection to an IMS database instance running remotely outside of the cloud infrastructure. The configuration properties allow a connection to be made to the IMS DB system.

**Before you begin**

The following are the attributes for an IMS database component:

- **Resource adapter**: Specifies the full path name of the IMS database system resource adapter. This attribute is required.
- **Server host name or IP Address**: Specifies the host name or IP address of the IMS database system. This attribute is required.
- **Server Port number**: Specifies the port number of the existing IMS database system. This attribute is required.
- **Datastore name**: Specifies the name of the target IMS data store. This attribute is required.
- **Metadata URL**: Specifies the Java metadata class that provides the database view of the IMS database. This attribute is required.
- **User name**: Specifies the name that is used to access the existing IMS database system.
- **Password**: Specifies the password associated with the user name property.

The following are optional properties:

**Connections**

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files).</td>
<td>• Java Naming and Directory Interface (JNDI) name of the data source • Resource references of the data source</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
<td>• JNDI name of the data source • Resource references of the data source</td>
</tr>
</tbody>
</table>

To make a connection between an application component and an existing IMS database, hover over the blue circle on the IMS database component part on the canvas. When the blue circle turns yellow, draw a connection between the IMS database and the application component. The application can use JNDI or Resource references settings to locate the data source. Specify the JNDI name or the Resource Reference in the link property panel. The name must match the name that is coded into the application.

**About this task**

You can view, edit, or add this virtual application component in the user interface as follows:
Procedure
1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To add a new IMS database component to a virtual application pattern, click the Existing IMS Database component listed under the Databases Components and drag the icon to the Pattern Builder canvas. The properties panel for the database component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To edit an existing IMS database component, select the Existing IMS Database component part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, see the properties descriptions or view the help by selecting the help icon on the properties panel.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the database component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select imsd/bb.x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results
You have edited a current component, edited an existing component, or added one.

User Registry components:

There are several user registry components to choose from when building a virtual application pattern.

About this task

• “Existing User Registry (IBM Tivoli Directory Server)”
• “Existing User Registry (Microsoft Active Directory)” on page 522
• “User Registry (Tivoli Directory Server)” on page 525

Existing User Registry (IBM Tivoli Directory Server):

An existing user registry cloud component represents an existing Lightweight Directory Access Protocol (LDAP) service that can be attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security.

Before you begin

The following are attributes for the user registry component:
• Server Hostname or IP address: Specifies the hostname or IP address of the remote LDAP. This attribute is required.
• Server Port Number: Specifies the port number of the remote LDAP. The default is 389 or 636 for Secure Socket Layer (SSL). This attribute is required.
• Login domain name (DN): Specifies the login DN. This attribute is required.
• **Password**: Specifies the password to access the remote LDAP. This attribute is required.

• **Domain Suffix of LDAP**: Specifies the domain suffix of the remote LDAP. This attribute is required.

• **Server SSL certificate**: Specifies the SSL port certificate for the remote LDAP.

• **User filter**: Specifies the LDAP user filter that searches the existing user registry for users.

• **Group filter**: Specifies the LDAP group filter that searches the existing user registry for groups.

**Default settings**

Tivoli Directory Server is registered to the federated repository in WebSphere Application Server using Virtual Member Manager (VMM) with the following settings:

• Login properties of VMM = uid or cn

• Entity type
  – Object class of PersonAccount = Person or inetOrgPerson in ITDS
  – Object class of Group = groupOfUniqueNames or groupOfNames in ITDS

  **Note**: The default value for the object class is groupOfUniqueNames. This value cannot be changed.

**Connections**

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Web application (WebSphere Application Server) | A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files). | • User filter  
• Group filter  
• Role name  
• User role mapping  
• Group role mapping  
• Special subject mapping |
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files). | • User filter  
• Group filter  
• Role name  
• User role mapping  
• Group role mapping  
• Special subject mapping |
| OSGi application (WebSphere Application Server) | OSGi application on WebSphere Application Server | • User filter  
• Group filter  
• Role name  
• User role mapping  
• Group role mapping  
• Special subject mapping |

To make a connection between an application component and an existing Tivoli Directory Server user registry, hover over the blue circle on the existing user registry component part on the canvas. When the blue circle turns yellow, draw a
About this task

The current implementation supports a one-time upload of users and groups in an LDAP Data Interchange Format (LDIF) file, and applications are currently limited to enterprise applications. Within the application, the roles are defined in the web.xml file. Bindings of roles to users and groups are defined in the META-INF/ibm-application-bnd.xml file. Bind the roles to group for ease of management.

You can view, edit, or add this virtual application component in the user interface as follows:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing user registry component, select the component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add an existing registry component to a virtual application pattern, click the Existing User Registry (IBM Tivoli Directory Server) component located under the User Registry Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the user registry component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select wasldap/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

Example

The following examples illustrate the three metadata files that are required to set up an enterprise application with the user registry component.

The LDIF file defines the users and groups for the application. user2 is in the group1 group.

dn: o=acme,c=us
objectclass: organization
objectclass: top
o: ACME

dn: cn=user2,o=acme,c=us
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top

About this task

The current implementation supports a one-time upload of users and groups in an LDAP Data Interchange Format (LDIF) file, and applications are currently limited to enterprise applications. Within the application, the roles are defined in the web.xml file. Bindings of roles to users and groups are defined in the META-INF/ibm-application-bnd.xml file. Bind the roles to group for ease of management.

You can view, edit, or add this virtual application component in the user interface as follows:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing user registry component, select the component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add an existing registry component to a virtual application pattern, click the Existing User Registry (IBM Tivoli Directory Server) component located under the User Registry Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the user registry component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select wasldap/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

Example

The following examples illustrate the three metadata files that are required to set up an enterprise application with the user registry component.

The LDIF file defines the users and groups for the application. user2 is in the group1 group.

dn: o=acme,c=us
objectclass: organization
objectclass: top
o: ACME

dn: cn=user2,o=acme,c=us
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
The binding file binds the group1 group to the role1 role.

The web.xml file defines the roles and security policy for the application. role1 can only access the protected resources.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <display-name>HitCountWeb</display-name>
  <servlet>
    <description></description>
    <display-name>HitCountServlet</display-name>
    <servlet-name>HitCountServlet</servlet-name>
    <servlet-class>com.ibm.samples.hitcount.HitCountServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HitCountServlet</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
  <security-constraint>
    <display-name>AllAuthenticated</display-name>
    <web-resource-collection>
      <web-resource-name>All</web-resource-name>
      <url-pattern>/*</url-pattern>
      <http-method>GET</http-method>
      <http-method>PUT</http-method>
      <http-method>HEAD</http-method>
      <http-method>TRACE</http-method>
      <http-method>POST</http-method>
      <http-method>DELETE</http-method>
      <http-method>OPTIONS</http-method>
    </web-resource-collection>
    <auth-constraint>
      <description>Auto generated Authorization Constraint</description>
      <role-name>role1</role-name>
    </auth-constraint>
    <user-data-constraint>
      <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
    <login-config>
      <auth-method>FORM</auth-method>
      <realm-name></realm-name>
    </login-config>
  </security-constraint>
  <login-config>
    <auth-method>FORM</auth-method>
    <realm-name></realm-name>
  </login-config>
</web-app>
```
Existing User Registry (Microsoft Active Directory):

An existing user registry cloud component represents an existing Lightweight Directory Access Protocol (LDAP) service that can be attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security.

Before you begin

The following are attributes for the user registry component:

- **Server Hostname or IP Address**: Specifies the host name or IP address of the remote LDAP Server. This attribute is required.
- **Server Port Number**: Port number of the remote LDAP. The default is 389 or 636 for Secure Sockets Layer (SSL). This attribute is required.
- **Login domain name (DN)**: Specifies the login DN. This attribute is required.
- **Password**: Specifies the password to access the remote LDAP. This attribute is required.
- **Domain suffix of LDAP**: Specifies the domain suffix of the remote LDAP. This attribute is required.
- **Server SSL certificate**: Specifies the SSL port certificate for the remote LDAP.
- **User Filter**: Specifies the LDAP user filter that searches the existing user registry for users.
- **Group Filter**: Specifies the LDAP group filter that searches the existing user registry for groups.

Default settings

Microsoft Active Directory Server is registered to the federated repository in WebSphere Application Server using Virtual Member Manager (VMM) with the following settings:

- Login properties of VMM = uid or cn
  - *samAccountName is mapped to both uid and cn
- Entity type
  - Object class of PersonAccount = user
  - Object class of Group = group

Connections
Table 57. Incoming connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Web application (WebSphere Application Server) | A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web archive (WAR files). | • Role name  
• User role mapping  
• Group role mapping  
• Mapping special subjects |
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise archive (EAR files). | • Role name  
• User role mapping  
• Group role mapping  
• Mapping special subjects |
| OSGi application (WebSphere Application Server) | OSGi application on WebSphere Application Server | • Role name  
• User role mapping  
• Group role mapping  
• Mapping special subjects |

To make a connection between an application component and an existing Microsoft Active Directory, hover over the blue circle on the user registry component part on the canvas. When the blue circle turns yellow, draw a connection between the user registry and component.

About this task

The current implementation supports a one-time upload of users and groups in an LDAP Data Interchange Format (LDIF) file, and applications are currently limited to enterprise applications. Within the application, the roles are defined in the web.xml file. Bindings of roles to users and groups are defined in the META-INF/ibm-application-bnd.xml file. Bind the roles to group for ease of management.

You can view, edit, or add this virtual application component in the user interface as follows:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing user registry component, select the component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add an existing Microsoft Active Directory user registry component to a virtual application pattern, click Existing User Registry (Microsoft Active Directory) listed under the User Registry Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the user registry component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select wasldap/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

Example

The following examples illustrate the three metadata files that are required to set up an enterprise application with the user registry component.

The LDIF file defines the users and groups for the application. user2 is in the group1 group.

```
dn: o=acme,c=us
objectclass: organization
objectclass: top
o: ACME

dn: cn=user2,o=acme,c=us
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
objectclass: ePerson
cn: user2
userpassword: user2
initials: user2
sn: user2
uid: user2
```

```
dn: cn=group1,o=acme,c=us
objectclass: groupOfNames
objectclass: top
cn: manager
member: cn=user2,o=acme,c=us
```

The web.xml file defines the roles and security policy for the application. role1 can only access the protected resources.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <display-name>HitCountWeb</display-name>
  <servlet>
    <description></description>
    <display-name>HitCountServlet</display-name>
    <servlet-name>HitCountServlet</servlet-name>
    <servlet-class>com.ibm.samples.hitcount.HitCountServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HitCountServlet</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
  <security-constraint>
    <display-name>AllAuthenticated</display-name>
    <web-resource-collection>
      <web-resource-name>All</web-resource-name>
      <url-pattern>/</url-pattern>
    </web-resource-collection>
```

524 IBM Cloud Orchestrator 2.4: User’s Guide
<http-method>GET</http-method>
<http-method>PUT</http-method>
<http-method>HEAD</http-method>
<http-method>TRACE</http-method>
<http-method>POST</http-method>
<http-method>DELETE</http-method>
<http-method>OPTIONS</http-method>

The binding file binds the group1 group to the role1 role.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<security-role name="role1">
  <group name="group1" />
</security-role>
</application-bnd>
```

User Registry (Tivoli Directory Server):

A user registry (Tivoli Directory Server) cloud component represents a pattern-deployed Lightweight Directory Access Protocol (LDAP) service that can be deployed by itself or attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security.

Before you begin

The following are attributes for the user registry component:

- **Base domain name (DN):** Specifies the base DN. This attribute is required.
- **LDIF file:** Specifies the name of the LDAP Data Interchange Format (LDIF) file. This attribute is required.
- **Custom schema file:** Specifies the name of the custom schema file.
- **User filter:** Specifies the LDAP user filter that searches the existing user registry for users. This attribute is required.
Group filter: Specifies the LDAP group filter that searches the existing user registry for groups. This attribute is required.

Default settings

Tivoli Directory Server is registered to the federated repository in WebSphere Application Server using Virtual Member Manager (VMM) with the following settings:
- Login properties of VMM = uid or cn
- Entity type
  - Object class of PersonAccount = Person or inetOrgPerson in ITDS
  - Object class of Group = groupOfUniqueNames or groupOfNames in ITDS

  Note: The default value for the object class is groupOfUniqueNames. This value cannot be changed.

Connections

Table 58. Incoming connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files).</td>
<td>• User filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Group filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Role name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• User role mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Group role mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Special subject mapping</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
<td>• User filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Group filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Role name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• User role mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Group role mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Special subject mapping</td>
</tr>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server</td>
<td>• User filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Group filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Role name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• User role mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Group role mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Special subject mapping</td>
</tr>
</tbody>
</table>

To make a connection between a component and the user registry, hover over the blue circle on the user registry component part on the canvas. When the blue circle turns yellow, draw a connection between the user registry and component.

About this task

The current implementation supports a one-time upload of users and groups in an LDIF file, and applications are currently limited to enterprise applications. Within the application, the roles are defined in the web.xml file. Bindings of roles to users and groups are defined in the META-INF/ibm-application-bnd.xml file. Bind the roles to group for ease of management.
You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing user registry component, select the component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a user registry component to a virtual application pattern, click User Registry (Tivoli Directory Server) listed under the User Registry Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the user registry component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select tds/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.

**Example**

The following examples illustrate the three metadata files that are required to set up an enterprise application with the user registry component.

The LDIF file defines the users and groups for the application. user2 is in the group1 group.

```l捻"f
dn: o=acme,c=us
objectclass: organization
objectclass: top
o: ACME

dn: cn=user2, o=acme, c=us
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
objectclass: ePerson
cn: user2
userpassword: user2
initials: user2
sn: user2
uid: user2

dn: cn=group1, o=acme, c=us
objectclass: groupOfNames
objectclass: top
.cn: manager
member: cn=user2, o=acme, c=us
```
The web.xml file defines the roles and security policy for the application. role1 can only access the protected resources.

```xml
<web-app id="WebApp_ID" version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
<display-name>HitCountWeb</display-name>
<servlet>
<description/>
<display-name>HitCountServlet</display-name>
<servlet-name>HitCountServlet</servlet-name>
<servlet-class>com.ibm.samples.hitcount.HitCountServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>HitCountServlet</servlet-name>
<url-pattern>/*</url-pattern>
</servlet-mapping>
<security-constraint>
<display-name>AllAuthenticated</display-name>
<web-resource-collection>
<web-resource-name>All</web-resource-name>
<url-pattern>/*</url-pattern>
<http-method>GET</http-method>
<http-method>PUT</http-method>
<http-method>HEAD</http-method>
<http-method>TRACE</http-method>
<http-method>POST</http-method>
<http-method>DELETE</http-method>
<http-method>OPTIONS</http-method>
</web-resource-collection>
<auth-constraint>
<description>Auto generated Authorization Constraint</description>
<role-name>role1</role-name>
</auth-constraint>
<user-data-constraint>
<transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
</security-constraint>
<login-config>
<auth-method>FORM</auth-method>
<realm-name></realm-name>
<form-login-config>
<form-login-page>/login.jsp</form-login-page>
<form-error-page>/login.jsp?error=Invalid+username+or+password</form-error-page>
</form-login-config>
</login-config>
<security-role>
<description>allowed group</description>
<role-name>role1</role-name>
</security-role>
</web-app>

The binding file binds the group1 group to the role1 role.

```xml
<security-role name="role1">
<group name="group1" />
</security-role>
</application-bnd>
Messaging components:

There are several message components to choose from when building a virtual application pattern.

About this task

- “Existing Messaging Service (WebSphere MQ)”
- “Existing topic (WebSphere MQ)” on page 531
- “Existing queue (WebSphere MQ)” on page 532

Existing Messaging Service (WebSphere MQ):

An existing message service component represents a connection to an external messaging system such as WebSphere MQ. The presence of a messaging system allows an enterprise application running on WebSphere Application Server to connect to the external messaging resource, such as WebSphere MQ.

Before you begin

The following are attributes for the messaging service:

- **Queue manager name**: Specifies the name of the queue manager to connect to. This attribute is required.
- **Server host name or IP address**: Specifies the TCP/IP host name or address of the external WebSphere MQ messaging service. This attribute is required.
- **Server Port Number**: Specifies the TCP/IP port on which the external WebSphere MQ message service is listening for connections. This attribute is required. The default port is 1414.
- **Channel name**: Specifies the name of the channel definition to use when accessing the WebSphere MQ queue manager. This attribute is required. The default is SYSTEM.DEF.SVRCONN.

Connections

**Table 59. Incoming connectable components**

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Web application (WebSphere Application Server) | A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files). | • Java Naming Directory Interface (JNDI) name of JMS connection factory  
  • Resource references of JMS connection factory  
  • Client ID |
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files). | • JNDI Name of JMS connection factory  
  • Resource references of JMS connection factory  
  • Client ID |
| OSGi application (WebSphere Application Server) | OSGi application on WebSphere Application Server | • JNDI Name of JMS connection factory  
  • Resource references of JMS connection factory  
  • Client ID |
The application is assumed to use JNDI settings to locate the topic. Specify the JNDI name in the link property panel, either as a hard-coded JNDI name or by selecting the relevant application resource-references from the property panel list box. During deployment, the JNDI name is set to the corresponding topic, and mapped, if required, to the relevant resource reference in the application.

To make a connection between an application component and the messaging service, hover over the blue circle on the messaging service component part on the canvas. When the blue circle turns yellow, draw a connection between the messaging service and application component.

About this task

The messaging service component represents a connection to an instance of IBM WebSphere MQ. The component can be configured to create a connection to your IBM WebSphere MQ installation. When you click the messaging service component on the Pattern Builder canvas, a properties panel displays.

You can view, edit, or add this virtual application component in the user interface as follows:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing messaging service component, select the component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a messaging service component to a virtual application pattern, click Existing Messaging Service (WebSphere MQ) listed under the Messaging Components and drag the icon to the Pattern Builder canvas. The properties panel for the transaction processing component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the existing WebSphere MQ service properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select wasmq/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.
Existing topic (WebSphere MQ):

A topic represents a message destination on an external WebSphere MQ messaging service through which messages are published and subscribed.

Before you begin

The following is a required attribute for topic:

- **Existing topic name**: Specifies the name of the topic destination on the external WebSphere MQ messaging service through which messages are published and subscribed.

Connections

*Table 60. Incoming connectable components*

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files).</td>
<td>• JNDI name&lt;br&gt;• Resource environment references&lt;br&gt;• Message destination references</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
<td>• JNDI name&lt;br&gt;• Resource environment references&lt;br&gt;• Message destination references</td>
</tr>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server</td>
<td>• JNDI name&lt;br&gt;• Resource environment references&lt;br&gt;• Message destination references</td>
</tr>
</tbody>
</table>

The application is assumed to use JNDI settings to locate the topic. Specify the JNDI name in the link property panel, either as a hard-coded JNDI name or by selecting the relevant application resource-references from the property panel list box. During deployment, the JNDI name is set to the corresponding topic, and mapped, if required, to the relevant resource reference in the application.

The required attributes for Link to WebSphere MQ topic are as follows:

- **JNDI name**: The JNDI name that the application uses to locate the topic destination. The JNDI name is only required if the application accesses the topic directly without using a resource environment reference or message destination reference.

- **Resource environment references**: The resource environment references that the application uses to locate the topic.

- **Message destination references**: The message destination references that the application uses to locate the topic.

To make a connection between a component and the messaging topic, hover over the blue circle on the topic component part on the canvas. When the blue circle turns yellow, draw a connection between the topic and component.
About this task

You can view, edit, or add this virtual application component in the user interface as follows:

Procedure

1. Click PATTERN > Pattern Design > Virtual Application Patterns.
2. Select a *virtual application pattern*.
3. Click *Edit the virtual application* icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing topic, select the *Existing Topic* component part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add an existing topic component to a virtual application pattern, click the *Existing Topic* component listed under the *Messaging Components* component and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the topic component properties by viewing the plug-in information. Click PATTERN > Deployer Configuration > System Plug-ins. Select `wasmqt/x.x.x.x` from the System Plug-ins palette where `x.x.x.x` corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

*Existing queue (WebSphere MQ):*

An existing message queue is a message queue on an external WebSphere MQ service from which messages are sent and received.

**Before you begin**

The following is a required attribute for a queue:

- **Existing queue name:** Specifies the name of the message queue on the external WebSphere MQ messaging service through which messages are sent and received.

**Connections**

*Table 61. Incoming connectable components*

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files).</td>
<td>JNDI name, Resource environment references, Message destination references</td>
</tr>
</tbody>
</table>
Table 61. Incoming connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files). | • JNDI name  
• Resource environment references  
• Message destination references |
| OSGi application (WebSphere Application Server)      | OSGi application on WebSphere Application Server                             | • JNDI name  
• Resource environment references  
• Message destination references |

The application is assumed to use JNDI settings to locate the queue. Specify the JNDI name in the link property panel, either as a hard-coded JNDI name or by selecting the relevant application resource-references from the property panel list box. During deployment, the JNDI name is set to the corresponding queue, and mapped if required to the relevant resource reference in the application.

The required attributes for Link to WebSphere MQ queue are as follows:

- **JNDI name**: The JNDI name that the application uses to locate the queue destination. This is only required if the application accesses the queue directly without using a resource environment reference or message destination reference.
- **Resource environment references**: The resource environment references that the application uses to locate the Queue.
- **Message destination references**: The message destination references that the application uses to locate the Queue.

To make a connection between a component and the messaging queue, hover over the blue circle on the queue component part on the canvas. When the blue circle turns yellow, draw a connection between the queue and component.

**About this task**

You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing queue, select the Existing Queue component part on the Pattern Builder canvas. The properties panel displays.
   For more details on these the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new queue component to a virtual application pattern, click the Existing Queue (WebSphere MQ) component listed under the Messaging Components and drag the icon to the Pattern Builder canvas. The properties
panel for the component displays to the right of the Pattern Builder palette. For
more details on the properties panel settings, view the help by selecting the
help icon on the properties panel.

6. You can also view the queue component properties by viewing the plug-in
information. Click PATTERNS > Deployer Configuration > System Plug-ins.
Select wasmq2/x.x.x.x from the System Plug-ins palette where x.x.x.x
corresponds to the version numbers. The component plug-in configuration
information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

OSGi components:

The OSGi components available as parts for your virtual application pattern are
the OSGi application and the external OSGi bundle repository.

About this task

- “Existing OSGi bundle repository (WebSphere Application Server)”
- “OSGi application (WebSphere Application Server)” on page 535

Existing OSGi bundle repository (WebSphere Application Server):

This component provides the URL of an existing WebSphere Application Server
OSGi bundle repository.

Before you begin

The following are the attributes for the external OSGi bundle repository:

- Bundle repository URL: Specifies the URL of the existing OSGi bundle
  repository. This attribute is required.

Connections

Table 62. Incoming connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server</td>
</tr>
</tbody>
</table>

To make a connection between an OSGi application component and the external
OSGi bundle repository, hover over the blue circle on the external OSGi bundle
repository component part on the canvas. When the blue circle turns yellow, draw
a connection between the external OSGi bundle repository and OSGi application
component.

About this task

You can view, edit, or add this virtual application component in the user interface
as follows:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click **Edit the virtual application** icon located in the upper right corner of the Pattern Builder palette.

4. To edit an existing external OSGi bundle repository component, select the **External OSGi Bundle Repository** component part on the Pattern Builder canvas. The properties panel displays.

   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.

5. To add an external OSGi bundle repository component to a virtual application pattern, click the **External OSGi Bundle Repository** component listed under the **OSGi Components** and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.

6. You can also view the external OSGi bundle repository component properties by viewing the plug-in information. Click PATTERNNS > **Deployer Configuration** > **System Plug-ins**. Select osgirepo/xx.x.xx from the **System Plug-ins** palette where xx.x.xx corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

**OSGi application (WebSphere Application Server):**

This component represents the OSGi application on WebSphere Application Server.

**Before you begin**

The following are attributes for the OSGi application component:

- **EBA file:** Specifies the OSGi application to be uploaded. The OSGi application is an enterprise bundle archive (EBA) file (.eba). This attribute is required.

Connections

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling policy (web or enterprise application)</td>
<td>Scaling is a workload pattern runtime capability to automatically scale your application platform as the load changes. A scaling policy component defines this capability and the conditions under which scaling activities are performed for your application.</td>
</tr>
<tr>
<td>Routing policy (web, enterprise, or OSGi EBA application)</td>
<td>Specifies a routing policy for a web application, enterprise application, or an OSGi EBA application.</td>
</tr>
<tr>
<td>Log policy (web or enterprise application)</td>
<td>A policy to specify configuration for log file records.</td>
</tr>
<tr>
<td>JVM policy (web or enterprise application)</td>
<td>A policy to control features of the underlying Java Virtual Machine (JVM).</td>
</tr>
<tr>
<td>Component name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Existing Topic (WebSphere MQ)</td>
<td>An existing topic represents a message destination on an external IBM WebSphere MQ messaging service through which messages are published and subscribed.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Existing Messaging service</td>
<td>An existing messaging service represents a connection to an external messaging system such as WebSphere MQ.</td>
</tr>
<tr>
<td>(WebSphere MQ)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Existing Database (Oracle)</td>
<td>An existing Oracle database component represents a connection to an existing Oracle database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote Oracle database.</td>
</tr>
<tr>
<td>Generic target</td>
<td>A component used to open the firewall for outbound TCP connections from a web or enterprise application to a specified host and port.</td>
</tr>
<tr>
<td>Database (DB2)</td>
<td>A database (DB2) component that represents a pattern-deployed database service.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Existing Database (DB2)</td>
<td>An existing DB2 database component represents a connection to a remote DB2 database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote DB2 database.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 64. Outgoing connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection</th>
</tr>
</thead>
</table>
| Existing Database (Informix)          | An existing Informix database component represents a connection to a remote Informix database instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the remote Informix database. | • JNDI name of the data source  
• Resource references of the data source  
• Non-transactional data source                                      |
| Existing CICS Transaction Gateway (TG)| An existing CICS TG component represents an existing connection to a CICSTG instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the CICS TG. | • JNDI name of the CICS TG resource                                                                                       |
| Existing User Registry (Tivoli Directory Server) | An existing user registry (Tivoli Directory Server) cloud component represents an existing Lightweight Directory Access Protocol (LDAP) service that can be deployed by itself or attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security. | • User filter  
• Group filter  
• Role name  
• User role mapping  
• Group role mapping  
• Special subject mapping                                      |
| Existing User Registry (Microsoft Active Directory) | An existing user registry (LDAP) cloud component represents an existing LDAP service that can be attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security. |                                                                                                                                                 |
| User Registry (Tivoli Directory Server) | A user registry (Tivoli Directory Server) cloud component represents a pattern-deployed LDAP service that can be deployed by itself or attached to a web application component or an enterprise application component. The LDAP service provides a user registry for container-managed security. |                                                                                                                                                 |
| Existing OSGi Bundle Repository       | The URL of an existing OSGi bundle repository.                                                                                                                                                           |                                                                                                 |
Table 64. Outgoing connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing Queue (WebSphere MQ)</td>
<td>A message queue on an external WebSphere MQ messaging service through which messages are sent and received.</td>
<td>• JNDI name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Resource environment references</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Message destination references</td>
</tr>
</tbody>
</table>

**Attention:** You can upload an .eba file to replace an OSGi application in the Virtual Application Console, but you cannot rename the archive as a part of the update.

To make a connection between a component and the OSGi application, hover over the blue circle on the OSGi application component part on the canvas. When the blue circle turns yellow, draw a connection between the OSGi application repository and component.

**About this task**

You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing OSGi application component, select the OSGi application component part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new OSGi application component to a virtual application pattern, click the OSGi Application component listed under the OSGi Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the OSGi application component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select was/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.
Transaction processing components:

There are several transaction processing components to choose from when building a virtual application pattern.

About this task
- “Existing CICS Transaction Gateway”
- “Existing IMS TM” on page 541

Existing CICS Transaction Gateway:

An existing CICS Transaction Gateway (TG) component represents a connection to an existing CICS TG instance running remotely outside of the cloud. The configuration properties allow a connection to be made to the CICS TG.

Before you begin

You must install a CICS TG resource adapter on IBM Cloud Orchestrator to be able to connect and use a CICS TG from within your cloud environment. See the topic, Installing a CICS resource adapter.

The following are attributes for the OSGi application component:
- **Connection URL**: Specifies the connection URL of the existing CICS TG. This attribute is required.
- **Port Number**: Specifies the port number of the existing CICS TG.
- **CICS Server Name**: Specifies the name of the CICS server to connect to.
- **CICS user ID**: Specifies the CICS user ID to be used if no other security credentials are available.
- **Password**: Specifies the password that is associated with the CICS user ID to be used.
- **SSL Keyring**: Specifies the Secure Socket Layers (SSL) keyring for securing a connection to an existing CICS TG, for example, a Java keystore file. Click Browse to search for the SSL keyring file on your system.
- **SSL Keyring Password**: Specifies the password for the SSL keyring.
- **Applid**: Specifies the APPLID for applications using this connection.
- **Applid Qualifier**: Specifies the APPLID qualifier for applications using this connection.
- **Tran Name**: Specifies the transaction identifier for the mirror transaction placed in EIBTRNID by CICS.
- **TPN Name**: Specifies the transaction identifier of the CICS mirror transaction.
- **Enable trace**: Specifies if a trace is enabled for the existing CICS TG.

Connections

Table 65. Incoming connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Web application (WebSphere Application Server)      | A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files). | • JNDI Name of the JCA Connection Factory  
• Maximum number of connections to the CICS Transaction Gateway |
Table 65. Incoming connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files). | • JNDI Name of the JCA Connection Factory  
• Maximum number of connections to the CICS Transaction Gateway |
| OSGi application (WebSphere Application Server) | OSGi application on WebSphere Application Server | • JNDI Name of the JCA Connection Factory  
• Maximum number of connections to the CICS Transaction Gateway |

About this task

The CICS TG component represents a connection to an instance of CICS TG. The component can be configured to create a connection to your CICS TG installation. When you click the CICS TG component on the Pattern Builder canvas, a properties panel is displayed.

You can view, edit, or add this virtual application component in the user interface as follows:

Procedure

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**.
2. Select a **virtual_application_pattern**.
3. Click **Edit the virtual application** icon located in the upper right corner of the Pattern Builder palette.
4. To add a new transaction processing component to a virtual application pattern, click the **Existing CICS Transaction Gateway** component listed under the **Transaction Processing Components** and drag the icon to the Pattern Builder canvas. The properties panel for the transaction processing component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To edit an existing CICS TG component, select the **Existing CICS Transaction Gateway** component part on the Pattern Builder canvas. The properties panel displays.
   You must specify the connection URL that the resource adapter uses to communicate with CICS TG in the form protocol://address, and specify the port number for which CICS TG is listening. The other fields in the properties panel are optional. If you have configured SSL on CICS TG you must also enter the name of the SSL keyring file and the SSL keyring password that you have configured. Enter the full path name to the SSL keyring file in the **SSL keyring** field, for example, `/mykeys/jsse/keystore.jks`.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the transaction processing component properties by viewing the plug-in information. Click **PATTERNS > Deployer Configuration > System**
Plug-ins. Select wasctg/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

Installing the CICS resource adapter:

Before you can use the CICS Transaction Gateway (CICS TG) in the IBM Cloud Orchestrator, you must install a CICS TG resource adapter.

Before you begin

You can use the ECI adapter, cicseci.rar, or the ECI adapter with two-phase commit support, cicseciXA.rar. IBM Cloud Orchestrator does not provide EPI support. The resource adapters are specific to your release of CICS TG and the one you use depends on the platform that you are using and whether you require two-phase or single-phase commit. For further details about CICS TG resource adapters, see [Using the ECI resource adapters](#).

About this task

To install a CICS TG resource adapter, log on to IBM Cloud Orchestrator as an administrator. Upload the CICS TG resource adapter for your CICS TG installation. You can now use, and configure a CICS TG component.

Procedure

1. Click PATTERNS > Deployer Configuration > System Plug-ins. A configuration dialog box displays.
2. Browse for the resource adapter.
3. Click OK.

Results

You have uploaded a new resource adapter.

What to do next

Add the CICS TG component to a virtual application pattern.

Existing IMS TM:

An existing Information Management Systems Transaction Manager (IMS TM) component provides an enterprise or web application that is running on WebSphere Application Server to connect to and submit transactions to an existing IMS system running remotely outside of the cloud.

Before you begin

The configuration properties allow a connection to be made to the IMS TM system. The following are the required properties:

- **Resource Adapter**: Specifies the file path name of the IMS TM resource adapter (.rar file).
- **Server host name or IP Address**: Specifies the host name or IP address of IMS Connect. IMS Connect is the TCP/IP listener component of IMS.
- **Port number**: Specifies the TCP/IP port used by the target IMS Connect.
- **Datstore name**: Specifies the name of the target IMS system. This name must match the ID parameter of the datastore statement that is specified in the IMS Connect configuration member.

The following are optional properties:
- **User name**: Specifies the security authorization facility (SAF) user name that is used for connections created by the connection factory.
- **Password**: Specifies the password associated with the user name property
- **CM0Dedicated**: If checked (indicates True), dedicated persistent socket connections are generated. If cleared (indicates False), shareable persistent socket connections are generated. The default is **False**.
- **SSL Enabled**: Check to use SSL connection to IMS TM. If using SSL then the following parameters are required:
  - **SSL Encryption Type**: Specifies the encryption type: Strong or weak. This is related to the strength of the ciphers, that is, the key length. By default, the encryption type is set to weak.
  - **SSL Keystore Name**: Specifies the full file path name of the keystore. Private keys and their associated public keys certificates are stored in password-protected databases called keystores. An example of a keystore name is `c:\keystore\MyKeystore.ks`.
  - **SSL Keystore Password**: Specifies the password for the keystore.
  - **SSL TrustStore Name**: Specifies the full file path name of the truststore. A truststore file is a key database file that contains public keys or certificates.
  - **SSL TrustStore Password**: Specifies the password for the truststore.
  - **Trace Level**: Specifies the level of information that is traced. Here are the possible values:
    - 0: No tracing or logging occurs
    - 1: Only errors and exceptions are logged (default)
    - 2: Errors and exceptions plus the entry and exit of important methods are logged
    - 3: Errors and exceptions, the entry and exit of important methods, and the contents of buffers sent to and received from IMS Connect are logged.

Connections

_Table 66. Incoming connectable components_

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java EE web applications (WAR files).</td>
<td>- JNDI Name of the JCA Connection Factory, or Resource references mapping - Maximum number of connections to IMS TM - Connection timeout</td>
</tr>
</tbody>
</table>
Table 66. Incoming connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
<th>Connection properties</th>
</tr>
</thead>
</table>
| Enterprise application (WebSphere Application Server) | An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files). | • JNDI Name of the JCA Connection Factory, or  
• Resource references mapping  
• Maximum number of connections to IMS TM  
• Connection timeout |

| OSGi application (WebSphere Application Server) | OSGi application on WebSphere Application Server | • JNDI Name of the JCA Connection Factory, or  
• Resource references mapping  
• Maximum number of connections to IMS TM  
• Connection timeout |

**About this task**

The IMS TMRA component represents a connection to an instance of IMS TMRA. The component can be configured to create a connection to your IMS TM installation. When you click the IMS TMRA component on the Pattern Builder canvas, a properties panel is displayed. You can view, edit, or add this virtual application component in the user interface as follows:

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**.
2. Select a **virtual_application_pattern**.
3. Click **Edit the virtual application** icon located in the upper right corner of the Pattern Builder palette.
4. To add a new transaction processing component to a virtual application pattern, click the **Existing IMS TM** component listed under the **Transaction Processing Components** and drag the icon to the Pattern Builder canvas. The properties panel for the transaction processing component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To edit an existing web application component, select the **Existing IMS TM** component part on the Pattern Builder canvas. The properties panel displays. You must specify the resource adapter used to communicate with IMS TM, the host name or IP address of the IMS Connect component of IMS TM, the port number on which IMS Connect is listening and the datastore name of IMS as specified in the IMS Connect Configuration member. The other fields in the properties panel are optional. For more details on the properties panel settings, see the properties descriptions or view the help by selecting the help icon on the properties panel.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the transaction processing component properties by viewing the plug-in information. Click **PATTERNS > Deployer Configuration > System**
Plug-ins. Select instmra/x.x.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

Other components:

There are several other components to choose from when building a virtual application pattern.

About this task

- "Generic target"
- "Debug" on page 585

Generic target:

A generic target component is used to open the firewall for outbound TCP connections from a web or enterprise application to a specified host and port.

Before you begin

The following are the attributes for a generic target component:

- **Server (IP or IP/netmask):** Specifies the target server. This attribute is required.
- **Port:** Specifies the destination port on the target server. This attribute is required.

Connections

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files).</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
</tr>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server.</td>
</tr>
</tbody>
</table>

To make a connection between an application component and the generic target component, hover over the blue circle on the Informix remote database component part on the canvas. When the blue circle turns yellow, draw a connection between the generic target component and the application component.

About this task

You can view, edit, or add this virtual application component in the user interface as follows:
Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing generic target component, select the Generic target component part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new generic target component to a virtual application pattern, click the Generic target component listed under Other Components and drag the icon to the Pattern Builder canvas. The properties panel for the component displays to the right of the Pattern Builder palette. For more details on these the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the generic target component properties by viewing the plug-in information. Click PATTERNS > Deployer Configuration > System Plug-ins. Select connect/xx.x.x from the System Plug-ins palette where x.x.x.x corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

Results

You have edited a current component, edited an existing component, or added one.

Policies:

There are several policies to choose from when building a virtual application pattern.

About this task

- “Scaling policy”
- “Routing policy” on page 547
- “Log policy” on page 548
- “JVM policy” on page 549

Scaling policy:

Scaling is a Pattern Builder runtime capability to automatically scale your application platform as the load changes. A scaling policy component defines this capability and the conditions under which scaling activities are performed for your application.

Before you begin

The following are the attributes for a scaling policy:

- **Enable session caching**: Specifies whether to use the session caching function in your application.
- **Scaling Type**: Specifies the scaling type used. You can select **Static**, **CPU Based**, **Response Time Based**, or **Web to DB**. Depending on the selection, zero or more of the other attributes are valid.
- **Number of instances**: Specifies the number of cluster members that are hosting the web application. The default value is 2. Acceptable value range is 2 through 10. This attribute is required.

- **Instance number range of scaling in and out**: Specifies the scaling range for instance members that are hosting the web application. Acceptable value range is 1 through 50. This attribute is required.

- **Minimum time (in seconds) to trigger add or remove**: Specifies the time duration condition to start scaling activity. The default value is 120 seconds. Acceptable value range is 30 through 1800. This attribute is required.

- **Scaling in and out when CPU usage is out of threshold range (in percentage)**: Specifies the processor threshold condition to start scaling activity. When the average processor utilization of your application platform is out of this threshold range, your platform is scaled in or out. The default value is 20 - 80%. Acceptable values range 0 - 100%.

- **Scaling in and out when web response time is out of threshold range (in milliseconds)**: Specifies the web application response time condition to start scaling activity. When the web application response time is out of this threshold range, your platform is scaled in or out. The acceptable values range from 0 -1000 ms.

- **JDBC connections wait time is out of the threshold range (in milliseconds)**: Specifies the JDBC connection wait state to start scaling activity. When the JDBC connections wait time is out of this threshold range, your platform is scaled in or out. The acceptable values range from 0 -10000 ms.

- **JDBC connection pools usage is out of the threshold range (in percentage)**: Specifies JDBC connection pool usage to start scaling activity. When the JDBC connection usage is out of this threshold range, your platform is scaled in or out. The acceptable values range 0 - 100%.

**Note**: Due to OpenStack limitations, vertical scaling (increasing memory and CPU on running systems without stopping the service) is not supported by IBM Cloud Orchestrator. Only horizontal scaling (increasing the number of virtual machines to balance the workload) is supported when required by the scaling policy.

### Connections

**Table 68. Outgoing connectable components**

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files).</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
</tr>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server.</td>
</tr>
</tbody>
</table>

### About this task

You can view, edit, or add a scaling policy in the user interface as follows:
Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing scaling policy, select the Scaling Policy part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new scaling policy to a virtual application pattern, you can:
   a. Click the Add a policy icon located in the application component part on the canvas. Select Scaling Policy from the list of policies. The scaling policy displays as a part in your application component part.
   or
   b. Click the Add policy for application icon on the upper left side of the canvas. Select the Scaling Policy from the list of policies. The scaling policy displays on the Pattern Builder canvas. The policy is applied to all applicable components in the canvas.

Results

You have edited a current component, edited an existing policy, or added one.

Routing policy:

You can apply a routing policy to the application component parts of your virtual application pattern.

Before you begin

The following are the attributes for a routing policy:

- Virtual hostname: Specifies the name of the virtual host for the routing policy. This attribute is required.
- HTTP: Specifies support for HTTP schema with a routing policy.
- HTTPS: Specifies support for HTTPS schema with a routing policy.

Connections

Table 69. Outgoing connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java EE web applications (WAR files).</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
</tr>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server.</td>
</tr>
</tbody>
</table>
About this task

You can view, edit, or add a routing policy in the user interface as follows:

Procedure
1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing routing policy, select the Routing Policy part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new routing policy to a virtual application pattern, you can:
   • Click the Add a policy icon located in the application component part on the canvas. Select Routing Policy from the list of policies. The routing policy displays as a part in your application component part.
   OR
   • Click the Add policy for application icon on the top left side of the canvas. Select the Routing Policy from the list of policies. The routing policy displays on the Pattern Builder canvas. The policy is applied to all applicable components in the canvas.

Results

You have edited a current component, edited an existing policy, or added one.

Log policy:

A log policy can be added to your application component part to specify configurations for log records.

Before you begin

The following are the attributes for a log policy:
• Log detail levels: Specifies the usage of log levels to control which events are processed by Java logging.
• Additional Log Files or Directories to Monitor: Specifies a semicolon delimited list of directories or files to monitor. To specify that an entry is a directory, suffix the entry with a slash, for example, /var/log/myApplication/, or prefix it with the string, such as, dir:/var/log/myApplication. You can use an asterisk wildcard in the file-specification only, for example, /var/log/myApplication/*.*.log. Using the wild card like the following, /var/log/*/my.log, is invalid. Any directory specified is visible in the Log Viewer.

Connections

Table 70. Outgoing connectable components

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java Platform, Enterprise Edition (Java EE) web applications (WAR files).</td>
</tr>
</tbody>
</table>
Table 70. Outgoing connectable components (continued)

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
</tr>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server.</td>
</tr>
</tbody>
</table>

About this task

You can view, edit, or add a log policy in the user interface as follows:

Procedure
1. Click PATTERNS > Pattern Design > Virtual Application Patterns.
2. Select a virtual_application_pattern.
3. Click Edit the virtual application icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing log policy, select the Log Policy part on the Pattern Builder canvas. The properties panel displays.
   For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new log policy to a virtual application pattern, you can:
   - Click the Add policy for application icon located in the application component part on the canvas. Log Policy from the list of policies. The log policy displays as a part in your application component part.
   or
   - Click the Add policy for application icon on the upper left side of the canvas. Select the Log Policy from the list of policies. The log policy displays on the Pattern Builder canvas. The policy is applied to all applicable components in the canvas.

Results

You have edited a current component, edited an existing policy, or added one.

JVM policy:

A Java virtual machine (JVM) policy controls the underlying JVM. You can attach the JVM policy to debug WebSphere Application Server processes using an integrated development environment (IDE) like IBM Rational® Application Developer for WebSphere.

Before you begin

The following are the attributes for a JVM policy:
- **Minimum heap size**: Specifies the minimum heap size of the JVM specified size in megabytes (MB).
- **Maximum heap size**: Specifies the maximum heap size of the JVM specified size in megabytes (MB).
- **Enable debug**: Specifies if the JVM is in debug mode.
- **Debug port**: Specifies the port that the JVM listens on for remote connections.
- **Client (IP or IP/netmask)**: The IP address of the host that is being used to debug.
- **Client**: Specifies an optional address of the debug client. This setting is used to restrict source access to the debug port. Value is an IP address, for example, 1.2.3.4; or IP/netmask, for example, 1.2.0.0/255.255.0.0, which matches anything in the 1.2. network.
- **Enable verbose garbage collection**: Specifies if the JVM has garbage collection enabled.
- **Generic JVM arguments**:
  - **Bit level**: Specifies if the bit level is set to 32 bit or 64 bit.

**Connections**

*Table 71. Outgoing connectable components*

<table>
<thead>
<tr>
<th>Component name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web application (WebSphere Application Server)</td>
<td>A web application cloud component represents an execution service for Java EE web applications (WAR files).</td>
</tr>
<tr>
<td>Enterprise application (WebSphere Application Server)</td>
<td>An enterprise application (WebSphere Application Server) cloud component represents an execution service for Java EE enterprise applications (EAR files).</td>
</tr>
<tr>
<td>OSGi application (WebSphere Application Server)</td>
<td>OSGi application on WebSphere Application Server.</td>
</tr>
</tbody>
</table>

**About this task**

When you enable debugging, the JVM is started in the debug mode and is listening on the specified port. A debugger on any client machine can attach to the JVM by default. You can specify a client IP address or IP/netmask to restrict access to the JVM. A client IP address, such as 10.2.3.5, allows a specific client machine to debug. An IP/netmask, such as 10.2.3.5/255.255.0.0, allows any machine on the 10.2 network to attach to the JVM.

You can view, edit, or add a JVM policy in the user interface as follows:

**Procedure**

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**.
2. Select a **virtual_application_pattern**.
3. Click **Edit the virtual application** icon located in the upper right corner of the Pattern Builder palette.
4. To edit an existing JVM policy, select the **JVM Policy** part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To add a new JVM policy to a virtual application pattern, you can:
   - Click the **Add policy for application** icon located in the application component part on the canvas. **JVM Policy** from the list of policies. The JVM policy displays as a part in your application component part.
   - or
Click the **Add policy for application** icon on the upper left side of the canvas. Select the **JVM Policy** from the list of policies. The JVM policy displays on the Pattern Builder canvas. You can connect the policy to an application component part by hovering over the blue circle on the application component part. When the blue circle turns yellow, draw a connection between the application component and the policy.

**Results**

You have edited a current component, edited an existing policy, or added one.

**What to do next**

For more detailed information about using Rational Application Developer for WebSphere, see [Debugging applications](#) in the WebSphere Application Server Information Center.

You can optionally use the IBM Monitoring and Diagnostic Tools for Java - Health Center (Health Center) to assess the current status of a running Java application. Health Center continuous monitoring provides information that helps you to identify and resolve problems with applications.

In IBM Cloud Orchestrator, you can configure the IBM Monitoring and Diagnostic Tools for Java - Health Center using the following attributes in the JVM policy:

- **Enable Health Center**: Specifies to start the JVM with Health Center enabled. Health Center is not enabled by default.
- **Health Center port**: Specifies the port for which the Health Center agent listens for remote connections.
- **Health Center Client**: Specifies the IP address of the Health Center client. This is an optional setting.

Technical information regarding the IBM Monitoring and Diagnostic Tools for Java - Health Center is available at the following URL:


**Developing plug-ins**

Plug-ins define the components, links, and policies that you use in the Pattern Builder to create virtual application patterns, or extend existing virtual application patterns. This guide describes how to develop your own custom plug-ins. Custom plug-ins add behavior and function that users can exploit to enhance and customize the operation of their virtual applications.

**Before you begin**

Download the **Plug-in Development Kit (PDK)**. You can also download the PDK from the IBM Cloud Orchestrator user interface Welcome page.

**About this task**

Use the following steps to develop your own custom plug-ins. The plug-in can be developed in the Eclipse tool or the integrated development environment (IDE) of your choice.
Procedure
1. Define and package plug-in artifacts.
   a. Define the config.json file.

   The config.json file is the only required file in a plug-in. The name, version, and patterntypes elements are all required. The name element specifies the name of the plug-in, and the version element defines the version number of the plug-in. The patterntypes element specifies the pattern types with which the plug-in is associated. The following example is a WebSphere Application Server Community Edition plug-in that extends the IBM Web Application Pattern type (not released with the product):

   ```json
   {
     "name": "wasce",
     "version": "1.0.0.1",
     "patterntypes": {
       "secondary": {
         "*": "*
       }
     },
     "packages": {
       "WASCE": {
         "requires": {
           "arch": "x86_64",
           "memory": 512,
           "disk": 300
         },
         "parts": {
           "part": "parts/wasce.tgz",
           "parms": {
             "installDir": "/opt/wasce"
           }
         }
       },
       "WASCE_SCRIPTS": {
         "parts": {
           "part": "parts/wasce.scripts.tgz"
         }
       }
     }
   }
   ```

   This example defines most of the following common elements:
   - **patterntype** element:
     The value is specified as webapp.
     This means that the capabilities contributed by this plug-in are available when you create patterns from the Web Application Pattern (not released with the product). No primary element and a secondary element of "*" means it shows up in the Pattern Builder for all pattern types.
   - **requires** element:
     This element contains other elements that specify resource requirements of the plug-in.
     - **os**
       Specifies the operating system that the plug-in requires.
     - **arch**
       Specifies the virtual machine architecture that the plug-in requires. In the previous config.json sample code, the specified architecture is 64-bit, X86.
     - **cpu**
       Specifies the minimum processing capacity that is required for each package defined by your plug-in. The requires element specifies the
required attributes of the package, all parts and node-parts in it. For cpu it represents the total required resources of each type for all parts and node-parts in the package.

- memory
  Specifies the minimum memory requirement for each package defined by your plug-in. The requires element specifies the required attributes of the package, all parts and node-parts in it. For memory it represents the total required resources of each type for all parts and node-parts in the package.

- disk
  Specifies the minimum disk requirement for each package defined by your plug-in. The requires element specifies the required attributes of the package, all parts and node-parts in it. For disk it represents the total required resources of each type for all parts and node-parts in the package.

**Note:** During the provisioning process, IBM Cloud Orchestrator adds up the minimum CPU, memory, and disk values for each package, and provisions a virtual machine that meets the specified requirements.

- packages element:
  Defines the file packages with both the part and nodepart elements. The example plug-in provides two packages: WASCE and WASCE_SCRIPTS. The WASCE package contains the parts/wasce.tgz part file. This archive contains the wasce image - all the files that compose wasce. The binaries are required to install WebSphere Application Server Community Edition, and package it directly in the plug-in.

  There are other options for specifying the required binaries. You can define a file attribute and have administrators upload the required binaries after loading the plug-in in IBM Cloud Orchestrator. You can also link to a remote server that stores the required artifacts. The WASCE_SCRIPTS package provides the life cycle scripts to install the WASCE image to the desired location, to install the enterprise archive (EAR) or web archive (WAR) file, and to start the server.

2. Define configurable application model components.

   The web and enterprise application archive components are displayed in the Pattern Builder. Each component is specified in the metadata.json file that is located in the plugin/appmodel directory of the plug-in archive and plugin development project. The following example illustrates the JSON to define the web archive component:

```json
[
    {
        "id" : "WARCE",
        "label" : "Web Application (WebSphere Application Server Community Edition)",
        "description" : "A web application cloud component represents an execution service for Java EE Web applications (WAR files).",
        "type" : "component",
        "thumbnail" : "appmodel/images/WASCE.png",
        "image" : "appmodel/images/WASCE.png",
        "category" : "application",
        "attributes" : [
            {
                "id" : "archive",
                "label" : "WAR File",
                "description" : "Specifies the web application (*.war) to be uploaded.",
                "type" : "file",
                "required" : true,
            }
        ]
    }
]```
There is a similar stanza for the enterprise archive component for its downloadable archive.

The first type field of the listing is important. The value options for this field are component, link or policy, and this defines the type in the application model. The id of the component is WARCE. This can be any value as long as it is unique.

The category refers to the tab under which this component is shown on the palette in the Pattern Builder. The attributes array defines properties for the component that you are defining. You can see and are able to specify values for these properties when using this component in the Pattern Builder. Attribute types include file, string (shown here), number, boolean, array, and range.

3. Define a template to convert the visual model into a physical model.

Plug-ins must provide the knowledge and logic for how to implement, or realize, the deployment of the defined components. In the case of the next example, the meaning of how to deploy an enterprise or web application component must be specified. To do this, a single transform is provided that translates the application model derived from what users build in the Pattern Builder into a concrete topology.

The following example displays a Velocity template that represents a transformation of the component into a JSON object that represents a fragment of the overall topology document. Each component and link must have a transform. In our plug-in, the WARCE and EARCE components share the same transform template.

```json
{
  "vm-templates": [
    {
      "name" : "${prefix}-wasce",
      "packages" : [ "WASCE", "WASCE_SCRIPTS" ],
      "roles" : [
        {
          "plugin" : "$provider.PluginScope",
          "name" : "WASCE",
          "type" : "WASCE",
          "quorum" : 1,
          "external-uri" : [{"ENDPOINT":"http://{SERVER}:8080"}],
          "parms":{
            "ARCHIVE" : "$provider.generateArtifactPath( $applicationUrl,
               ${attributes.archive} )"
          },
          "requires" : { "memory":512, "disk":300 }
        },
        "scaling" : { "min":1, "max":1 }
      ]
    }
  ]
}
```

The topology fragment is a JSON object that contains a vm-templates element, which is an array of vm-templates. A vm-template is a virtual machine template, and defines the parts, nodeparts, and attributes of a virtual machine to be deployed. For this example, only a single vm-template that contains four important elements is needed:

- name: Specifies a unique name for a deployed virtual machine.
packages: Specifies a list of parts and nodeparts that are installed on each deployed virtual machine. The WASCE entry indicates the use of the WASCE virtual image. The WASCE_SCRIPTS entry specifies the WASCE life cycle scripts.

roles: Specifies parts in a plug-in that invoke lifecycle scripts for roles. You can have one or more roles in your plug-in, but in the sample plug-in there is a single WASCE role. When all roles on a node go to the RUNNING state, the node changes to the green RUNNING state.

4. Define lifecycle scripts to install, configure, and start software.

In this step, you define the lifecycle scripts for the plug-in. This process includes writing scripts to install, configure, and start the plug-in components. You can view the complete scripts in the downloadable archives. The following information includes the key artifacts:

* install.py script

The install.py script copies the WASCE image from the download location to the desired installDir folder. It also sets the installDir value in the environment for subsequent scripts. All parts and nodeparts installed by the IBM Cloud Orchestrator agent run as root. The chown -R virtuser:virtuser command changes file ownership of the installed contents to the desired user and group. Finally, the install.py script makes the scripts in the WebSphere Application Server Community Edition bin directory executable. The following sample code is the contents of the install.py script:

```python
installDir = maestro.parms['installDir']
maestro.trace_call(logger, ['mkdir', installDir])

if not 'WASCE' in maestro.node['parts']:
    maestro.node['parts']['WASCE'] = {}
    maestro.node['parts']['WASCE']['installDir'] = installDir

# copy files to installDir to install WASCE
this_file = inspect.currentframe().f_code.co_filename
this_dir = os.path.dirname(this_file)
rc = maestro.trace_call(logger, 'cp -r %s/files/* %s' % (this_dir, installDir), shell=True)
maestro.check_status(rc, 'wasce cp install error')

rc = maestro.trace_call(logger, ['chown', '-R', 'virtuser:virtuser', installDir])
maestro.check_status(rc, 'wasce chown install error')

# make shell scripts executable
rc = maestro.trace_call(logger, 'chmod +x %s/bin/*.sh' % installDir, shell=True)
maestro.check_status(rc, 'wasce chmod install error')
```

This example shows how the script makes use of the maestro module provided within the plug-in framework. The module provides several helper methods that are useful during installation and elsewhere.

* wasce.scripts part and install.py script

The wasce.scripts part also contains a install.py script. This script installs the WebSphere Application Server Community Edition life cycle scripts. The following is an example of the install.py script in wasce.scripts:

```python
# Prepare (chmod +x, dos2unix) and copy scripts to the agent scriptdir
maestro.install_scripts('scripts')
```

* configure.py script

The configure.py script in the wasce.scripts part installs the user-provided application to WebSphere Application Server Community Edition. The script takes advantage of the hot deploy capability of WebSphere Application Server Community Edition and copies the application binaries to a monitored directory. The following example includes the contents of the configure.py script:
The `start.py` script in the `wasce.scripts` part is responsible for starting the WebSphere Application Server Community Edition process. After starting the process, the script updates the state of the role to `RUNNING`. When the deployment is in the `RUNNING` state, you can access the deployed application environment. The following example shows the use of the `geronimo.sh start` command to start WebSphere Application Server Community Edition, as well as the `gsh.sh` command to wait on startup:

```
wait_file = os.path.join(maestro.node['scriptdir'], 'WASCE', 'wait-for-server.txt')
installDir = maestro.node['parts']['WASCE']['installDir']
rc = maestro.trace_call(logger, ['su', '-l', 'virtuser', installDir + '/bin/geronimo.sh', 'start'])
maestro.check_status(rc, 'WASCE start error')
logger.info('wait for WASCE server to start')
rc = maestro.trace_call(logger, ['su', '-l', 'virtuser', installDir + '/bin/gsh.sh', 'source', wait_file])
maestro.check_status(rc, 'wait for WASCE server to start error')
maestro.role_status = 'RUNNING'
logger.info('set WASCE role status to RUNNING')
logger.debug('Setup and start iptables')
maestro.firewall.open_tcpin(dport=1099)
maestro.firewall.open_tcpin(dport=8080)
maestro.firewall.open_tcpin(dport=8443)
```

There are other scripts and artifacts that make up the plug-in, but the above provides an explanation of the most significant scripts.

**What to do next**

Add your custom plug-in to IBM Cloud Orchestrator where the plug-in can be used to create or extend a virtual application.

**Plug-in Development Kit:**

The Plug-in Development Kit (PDK) is designed to help you build plug-ins for IBM Cloud Orchestrator. The custom plug-ins can be added to the IBM Cloud Orchestrator catalog where they are used to add components, links, and policies to virtual applications.

**Attention:** Download the [PDK](#) to get started.

The PDK is a zip package that includes a plug-in and pattern type build environment, samples, and a tool to create a plug-in starter project.

- docs
Contains documentation for the PDK
- docs/index.html Contains a list of links to the documentation in the docs
directory.
- docs/PDKSampleUsersGuide.pdf
  The PDK Samples User Guide. The Samples are also located in the
  information center.
- docs/javadoc
  This directory contains Javadoc for IBM Cloud Orchestrator interfaces that the
  plug-ins can invoke from the Java code.
- docs/pydoc
  This directory contains documentation for the maestro module used in
  lifecycle Python scripts for nodeparts and parts.

* iwd-pdk-workspace
  The root directory of your plugin development workspace.
  Each plug-in and pattern types has its own project directory in this root
directory. These directories can be used directly from the command line or
imported into Eclipse as plug-ins.
* pdk-debug-{version}.tgz
  This file is the debug plug-in that can be installed into the IBM Cloud
  Orchestrator instance and used to develop and debug the plug-ins.
  The debug includes features to deploy and debug a topology document, which
  is a JSON object, and debug plug-in installation and lifecycle scripts on
  deployed nodes. For more information, see the topic Debug.
* pdk-unlock-{version}.tgz
  The unlock plug-in enables you to delete a plug-in in use by a deployed
  application, replace it with an updated version, and activate the modified
  plug-in on deployed virtual machines in the application. For more information,
  see the topic Unlock.

**plugin.depends tool**

The plugin.depends tool is provided in the
IBMWorkloadPluginDevKit_<version>.zip file. This plug-in development tool is a
standard OSGi plug-in project. The tool includes IBM Cloud Orchestrator plug-in
libraries for development, build tools for plug-in and pattern types, and an Ant
build library, including:
- The lib folder that includes all of the Java archive (JAR) files that are required
  for plug-in development.
- The lib-build folder that includes all of the libraries that are required for the
  plug-in build script.
- The build/build.plugins.xml file that is the base internal build script file for
  single plug-in building. The build script file of each plug-in imports this build
  script file first and adds more actions, if necessary.
- The build/build.patterntypes.xml file which is a generic pattern type building
  script. The build script file of each pattern type imports this build script file first
  and adds more actions, if necessary.
- The create.plugin.project.xml file which is an Ant script used to create
  projects for the plug-ins in your workspace.
You can use the create.plugin.project.xml file of plugin.depends to create
projects for your plug-ins. This file creates a template or a Java plug-in project.
There are two required parameters: project-name and plugin-name. Using these
two parameters creates a template project. The third parameter, java.classname, is optional. If a valid class name is given, a Java plug-in project is created. The class name can be a simple name like MyPlugin or a package-qualified name, like com.acme.iwd.plugin.MyPlugin.

**Important:** Do not add the extension,.java to the end of the Java plug-in project, because the extension is assumed.

- The plugin-project-template that is used by the create.plugin.project.xml Ant script to create plug-in projects.

**Samples**

A Samples package is included with the PDK. See the topic, **Samples: Plug-in Development** for more information.

**Development plug-ins**

Several plug-ins to assist with plug-in development are included in the PDK. For more information, see “Plug-ins for development” on page 585.

**Installing the Plug-in Development Kit:**

To get started using the Plug-in Development Kit, first download and install the kit.

**Before you begin**

You can download the PDK from this IBM website or download the PDK from the IBM Cloud Orchestrator user interface Welcome page.

**Attention:** You must enable the PDK license before you can use the PDK. A dialogue box displays during download to assist you with the license acceptance process.

**About this task**

Use the following steps to download and install the PDK:

**Procedure**

1. Download the PDK from [IBM Plug-in Development Kit](https://www.ibm.com) or download the PDK from the IBM Cloud Orchestrator user interface Welcome page.
2. Extract the .zip file into a directory. From the command line, change to that directory and run Ant from the command line in that directory.

**Important:** You must accept the PDK license presented in a dialogue box to continue and unpack the contents of the PDK.

The following files are extracted into the file directory:

- **docs**
  - **docs/index.html** Contains a list of links to the documentation in the docs directory.
  - **docs/PDKSamplesUserGuide.pdf** The PDK Samples User Guide. The Samples are also located in the information center.
This directory contains Javadoc for IBM Cloud Orchestrator interfaces that the plug-ins can invoke from the Java code.

- docs/pydoc
  This directory contains documentation for the maestro module used in lifecycle Python scripts for nodeparts and parts.

- iwd-pdk-workspace
  The root directory of your plugin development workspace. Each plug-in and pattern types has its own project directory in this root directory. These directories can be used directly from the command line or imported into Eclipse as plug-ins.

- pdk-debug-{version}.tgz
  This file is the debug plug-in that can be installed into the IBM Cloud Orchestrator instance and used to develop and debug the plug-ins. The debug includes features to deploy and debug a topology document, which is a JSON object, and debug plug-in installation and lifecycle scripts on deployed nodes. For more information, see the topic, Debug.

- pdk-unlock-{version}.tgz
  The unlock plug-in enables you to delete a plug-in in use by a deployed application, replace it with an updated version, and activate the modified plug-in on deployed virtual machines in the application. For more information, see the topic, Unlock.

Results
The PDK is downloaded and installed. Now you must complete the task of “Setting up the plug-in development environment.”

Setting up the plug-in development environment:
Set up the environment to develop custom plug-ins that are used in IBM Cloud Orchestrator

Before you begin
The following products are required before setting up the environment:
  - Eclipse is not required, but if you use it, use this version.
  - If you use Eclipse, you can use the Ant that comes with it. Do not install Ant separately. Ant is located in the Eclipse installation directory at eclipse/plugins/org.apache.ant_1.*.
- Java Standard Edition (SE) 6, 32-bit
- Apache Ant, 1.7 or later

About this task
To set up your environment, complete the following steps:

Procedure
1. From the command line type cd iwd-pdk-workspace/plugin.depends.
2. In the plugin.depends project, run the build.xml Ant script. To run the Ant script, right-click on the file and select Run As > Ant Build. OR, type ant in the command line. This command builds all the plug-ins in the workspace.

3. Access the patterntypetest.basic project and run the build.patterntype.xml script. Type ant -f build.patterntype.xml. This command builds the pattern type.

4. Refresh the patterntypetest.basic project. A folder named, export, displays.

5. Navigate to the root of the export folder. The .tgz pattern type binary file is located here. The export/archive directory contains the built pattern type that is ready for installation into IBM Cloud Orchestrator.

6. Import the pattern type .tgz and use the plug-in from the IBM Cloud Orchestrator catalog.

Plug-in development guide:

If you are developing custom plug-ins, this topic provides more details about various aspects of plug-ins in the order encountered during a typical development effort.

The following list is the high-level sections for this plug-in development reference guide:

- **Kernel services**
  - Transformers: TopologyProvider services
  - Enhance template transforms with Java code
  - Plug-in components available as OSGi Declarative Services
- **Deployment** on page 566
  - Activation
  - Nodeparts
  - Parts
  - Roles
  - Set repeatable task
  - Recovery: Reboot or replace?
  - Virtual Application Console
- **Other reference** on page 581
- **Application model and topology document examples** on page 581

Kernel services

Transformers: TopologyProvider services

TopologyProvider implementations are plug-in-specific services for transforming component, links, and policies from an application model into an unresolved topology.

The transform step is a multi-step operation. First, the components are transformed. Attached policies are integrated into the component as extended attributes. Each component transformer takes the associated object from the application model as input, and returns a corresponding fragment of the topology document. The topology document and its fragments are JSON object documents. Links are transformed after the components. Links modify the component-generated topology documents, for example, parts and depends objects.
are added to the source roles. Each link transformer receives the topology fragments as input that is generated by the link source and target components. There are two types of transformers:

- Template-based implementations

Most transforms can be described using a template of the intended JSON document (topology fragment for components; depends objects for links). IBM Cloud Orchestrator embeds Apache Velocity 1.6.2 as a template engine. Template-based implementations include:

  - Component document

    The component name must match the "id" of the component, link, and policy that is defined in the plug-in appmodel/metadata.json file. Template files are specified as component properties, where the value is a path relative to the plug-in root. For example, the transformer for the sample starget component and link looks like the following:

    ```xml
    <?xml version="1.0" encoding="UTF-8"?>
    <scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="starget">
      <implementation class="com.ibm.maestro.model.transform.template.TemplateTransformer"/>
      <service>
        <provide interface="com.ibm.maestro.model.transform.TopologyProvider"/>
      </service>
      <property name="component.template" type="String" value="templates/starget_component.vm"/>
      <property name="link.template" type="String" value="templates/starget_link.vm"/>
    </scr:component>
    ```

  - Implementation

    The sample starget_component.vm illustrates component transformation as follows:

    ```json
    {
    "vm-templates": [
    {
      "scaling":{
        "min": 1,
        "max": 1,
      },
      "name": "${prefix}-starget",
      "roles": [
      {
        'parms': {
          "st1": "$attributes.st1"
        },
        "type": "starget",
        "name": "starget"
      }
    ]
    }
    ]
    }
    ```

    The sample starget_link.vm illustrates link transformation as follows:

    ```text
    ## Link templates render the depends objects to be added to the source role.
    ## sourceRole is required to locate the source of the link.
    ## Value is the type of the source role.
    #set( $sourceRole = "ssource" )
    
    ## sourcePackages is an optional array. Values in the array are added to the packages
    ## of the vm-template that is hosting the source role.
    #set( $sourcePackages = ["pkg2"] )
    
    ## Obtain a tuple related to the matching target role: target.template == vm-template
    ## that holds the target role; target.role == role
    ## String argument is the type of the target role.
    ```
## Validate target. If not found, throwHttpException

```
#set( $target = $provider.getMatchedRole($targetFragment, "starget") )

#if( $target == $null )
$provider.throwHttpException("Target Role starget not found.")
#end
```

```json
[
  {
    "role": "${target.template.name}.${target.role.name}",
    "type": "type",
    'parms': {
      "sl1": "$attributes.sl1"
    }
  }
]
```

The sample `ssource_component.vm` illustrates a more complex component transformation. The `#if_value` is a Velocimacro for conditional rendering of formatted strings: If `$map` contains a non-empty value for `$key`, then `$format_str` is evaluated and the value is available as `$value`.

```json
"vm-templates": [
  {
    "scaling":{
      "min": 1,
      "max": 1
    },
    "name": "${prefix}-ssource",
    "roles": [
      {
        'parms': {
          "sl1": "$attributes.sl1"
        }
      }
    ]
  }
]
```

## Handling optional attributes:

- **String value:**
  ```velocimacro```
  ```
  #if_value( $attributes, "ss_s", "ss_s": "$value", )
  ```
- **Number value:**
  ```velocimacro```
  ```
  #if_value( $attributes, "ss_n", "ss_n": $value, )
  ```
- **Boolean value:**
  ```velocimacro```
  ```
  #if_value( $attributes, "ss_b", "ss_b": $value, )
  ```
- **Missing value -- will not render:**
  ```velocimacro```
  ```
  #if_value( $attributes, "not_defined", "not_defined": $value, )
  ```

## Handling required attributes; throws an exception if the attribute is null/empty/not defined

- **String value:**
  ```velocimacro```
  ```
  "ss_f": "$provider.generateArtifactPath( $applicationUrl, ${attributes.ss_s} )",
  ```
- **Number value:**
  ```velocimacro```
  ```
  "ss_r_min": $attributes.ss_r.get(0),
  "ss_r_max": $attributes.ss_r.get(1)
  ```
- **Handling policies:**
  spolicy is defined; not_policy is not

```velocimacro```
```
#set( $spattrs = $provider.getPolicyAttributes($component, "spolicy") )
#if_value( $spattrs, "sp1", "sp1": $value, )
#if_value( $spattrs, "not_defined", "not_defined": $value, )
```
```velocimacro```
```
#set( $npattrs = $provider.getPolicyAttributes($component, "no_policy") )
#if_value( $npattrs, "np1", "np1": $value, )
```

## Handling required config parms; throws an exception if the parm is null/empty/not defined

- **String value:**
  ```velocimacro```
  ```
  "cp1": "$config.cp1"
  ```
- **Number value:**
  ```velocimacro```
  ```
  "cp2": $attributes.cp2
  ```

For artifacts, Inlet may send app model with absolute URLs for artifacts; other request paths might invoke with relative URLs. So use `generateArtifactPath()` function, which invokes `URI.resolve()` function that handles both cases.
Other available template features

Using static Java static classes

You can insert Java classes into the context with the $provider.getClassForName method. This feature is useful when static methods are used on these classes in your template. For example:

```java
#set( $Math = $provider.getClassForName("java.lang.Math") )
"ss_r_math_max":$Math.max($attributes.ss_r.get(0), $attributes.ss_r.get(1)),
```

`ss_r` is a range value, as defined in the plug-in appmodel/metadata.json file, which is a list with two long integer values. The lower range value is the first value in the list, with index 0. The upper range is the second value, with index 1. The previous example returns the upper range value, but is intended to show the usage of the java.lang.Math.max static method.

- Java implementations

For cases where templates are not sufficient, Java implementations can be used. Java implementations can generate the JSON documents with the included JSON APIs (com.ibm.json.java.*) or by modifying templates. Another option is to use templates and enhance them with Java functions. See the section, Enhancing template transforms with Java code, for details. This alternative is preferred.

- Component document

The component name must match the ID of the component, link, and policy that are defined in the plug-in appmodel/metadata.json file. For example, the transformer for the web archive (WAR) component is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0" name="WAR">
  <implementation class="com.ibm.maestro.model.transform.was.WARTransformer"/>
  <service>
    <provide interface="com.ibm.maestro.model.transform.TopologyProvider"/>
  </service>
</scr:component>
```

- Implementation

Implementations extend com.ibm.maestro.model.transform.TopologyProvider and can implement component and link transformations by overriding the corresponding methods:

```java
public JSONObject transformComponent(
  String vmTemplateNamePrefix,
  String applicationUrl, 
  JSONObject applicationComponent, 
  Transformer transformer) 
  throws Exception {
    return new JSONObject();
}
```

```java
public void transformLink(
  JSONObject sourceFragment, 
  JSONObject targetFragment, 
  String applicationUrl, 
  JSONObject applicationLink, 
  Transformer transformer) 
  throws Exception {
}
```

- Invoking templates
Java implementations can start templates using the following methods of TopologyProvider:

```java
public static JSONArtifact renderTemplateToJSON(Bundle b, String template, String logTag, Context context) throws HttpException;

public static String renderTemplate(Bundle b, String template, String logTag, Context context) throws HttpException;
```

For example, the WebSphere Application Server transformer starts a template to generate the topology fragment for a WebSphere Application Server instance as follows:

```java
protected void activate(ComponentContext context){
    _bundle = context.getBundleContext().getBundle();
}

@Override
public JSONObject transformComponent(String prefix, String applicationUrl, JSONObject component, Transformer transformer) throws Exception {
    JSONObject topology;
    JSONObject scalingPolicy = getPolicy(component, "ScalingPolicyofWAS");
    String vmTemplateName = prefix + "+"-was";

    if (scalingPolicy == null) {
        VelocityContext context = new VelocityContext();
        context.put(TemplateTransformer.PREFIX, prefix);
        context.put(TemplateTransformer.APPLICATION_URL, applicationUrl);
        context.put(TemplateTransformer.COMPONENT, component);
        JSONObject attributes = (JSONObject) component.get("attributes");
        context.put(TemplateTransformer.ATTRIBUTES, new RequiredMap(attributes));
        context.put(TemplateTransformer.CONFIG, new RequiredMap(getConfigParms()));
        context.put(TemplateTransformer.PROVIDER, this);
        String logTag = "WAS:templates/SingleWAS.vm";
        topology = (JSONObject) renderTemplateToJSON(_bundle, "templates/SingleWAS.vm", logTag, context);
    }
}
```

### Enhancing template transforms with Java code

Template transforms are recommended. If you need Java code, do as much as you can with the template, and then add Java methods that you start from your template as follows:

1. Create your Java class and have it extend TemplateTransformer.
2. Add your Java methods. The public methods can be started from your template by using $provider.myMethod(). You can pass parameters into the Java methods.
3. Update your OSGi component document to set the implementation class to your new Java class, not TemplateTransformer. There are two methods for updating the velocity context:
   - The velocity context is available in the velocity context, therefore you can pass it into a Java method using $context.
   - Implement the protected VelocityContext createContext(String applicationUrl, JSONObject component) method. In your method, call super.createContext(applicationUrl, component). The returned VelocityContext is the velocity context, to which you can add your custom objects.

   **Important:** Be careful not to overwrite any existing keys.
The following code example illustrates enhancing the templates:

**osgi.xml file:**
```xml
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/v.1.1.0" name="WASDB2">
<implementation class="com.ibm.maestro.model.transform.wasdb2.WASDB2LinkTransform"/>
<service>
<provide interface="com.ibm.maestro.model.transform.TopologyProvider"/>
</service>
<property name="link.template" type="String" value="templates/wasdb2_link.vm"/>
</scr:component>
```

**Java file:**
```java
package com.ibm.maestro.model.transform.wasdb2;

import com.ibm.maestro.model.transform.template.RequiredMap;
import com.ibm.maestro.model.transform.template.TemplateTransformer;

public class WASDB2LinkTransform extends TemplateTransformer {
    public static JndiNameResourceRefs getJndiNameAndResourceRefs(RequiredMap attributes)
            throws HttpException {
        return JndiNameResourceRefs.getJndiNameAndResourceRefs(attributes);
    }
}
```

**Plug-in components available as OSGi Declarative Services**

IBM Cloud Orchestrator provides limited support for OSGi services within plug-ins. Specifically, plug-ins can provide implementations of specific IBM Cloud Orchestrator service interfaces such as:

- **TopologyProvider**
  - Review the Javadoc for more information.

  **Attention:** Javadoc is included with the pdk.zip. Extract the javadoc.zip file and expand the file to get the Javadoc.

- **TopologyProcessor**
  - Review the Javadoc for more information.

- **ServiceProvisioner**
  - Review the Javadoc for more information.

- **PostProvisioner**
  - Review the Javadoc for more information.

  ```java
  package com.ibm.maestro.iaas;
  
  import org.apache.wink.client.RestClient;
  import com.ibm.json.java.JSONObject;
  
  public interface PostProvisioner {
    public void provisioned(final JSONObject topology, final JSONObject deployment, RestClient restClient);
  }
  ```

  Review the Javadoc for more information.

- **AppBindingsService**
- **RegistryProvider**
  - Review the Javadoc for more information.
Kernel services manage multiple versions of these services according to the plug-ins associated with the application model. However, version management does not apply to other services, so errors might occur if a plug-in exports another service implementation.

**Deployment**

**Activation**

Each image contains a startup script that is started after the virtual machine starts. This script is the hook point behaviors on a virtual machine, therefore, understanding the sequence of operations is helpful.

Each virtual machine is assigned a unique name within the deployment. The name is set as an environment variable named SERVER_NAME. The value is formed by appending an ordinal to the corresponding vm-template name, for example, application-was.1, application-was.2. Activation proceeds as follows:

1. Get this virtual machine vm-template from the topology document, for example, if SERVER_NAME == application-was.1, then get the vm-template named application-was.
2. For each nodepart in the vm-template:
   a. Download the nodepart .tgz file and extract into the `{nodepkgs_root}` directory.
   b. Invoke `{nodepkgs_root}/setup/setup.py`, if the script exists. Associated parms from the topology document are available as maestro.parms.
   c. Delete `{nodepkgs_root}/setup/`.
3. Run the installation scripts (`{nodepkgs_root}/common/install/*.sh|.py`) in ascending order.
4. Run the start scripts (`{nodepkgs_root}/common/start/*.sh|.py`) in ascending order.

In Step 2, nodeparts should not rely on the order of installation; that is, the setup/setup.py script should rely on contents of that nodepart only. One exception is the maestro module. The module initialization script is in place, so that the setup.py script can use the maestro HTTP client utility methods, for example, maestro.download(), and maestro.parms, to obtain configuration parameters.

Both installation and start scripts are ordered. By convention, these scripts are named with a number prefix, such as 5_autoscaling.sh or 9_agent.sh. These scripts are said to be in slot 5 or slot 9. All installation scripts in slot 0 are run before any installation script in slot 1. All of the installation scripts are run in numeric order, then all of the start scripts are run in numeric order. Set up and installation scripts are started one time for each virtual machine; start scripts are started on every boot or restart. For more information, see the section, ▲Recovery: Reboot or replace. The workload agent is packaged and installed as a nodepart.

**Nodeparts**

Nodeparts are installed by the activation script and generally contain binary and script files to augment the operating system. Review the following information about nodeparts:

- Conventions
Nodeparts are packaged as .tgz files. The contents are organized into a directory structure by convention. The following files and directories are optional:

- `common/python/maestro/{name}.py`
- `common/scripts/{name}`
- `common/start/{N}_{name}`
- `common/stop/{N}_{name}`
- `{name}/[any files]`
- `setup/setup.py`

The root directory is `common/` for the following shared files:

- `common/python`
  - Applies to the Python scripts started by the workload agent including:
    - `common/python` is added to the PYTHONPATH
    - All files in `common/python/maestro/*.py` are added to the maestro package
- `common/scripts`
  - Added to the PATH before the start scripts are run.
- `common/start`
  - Contains script files that are started automatically by the activation script; scripts are named with a number prefix, for example, `5_autoscaling.sh` or `9_agent.sh`, and run in numeric order starting with `0_*`.
- `common/stop`
  - Contains script files that are stopped automatically by the activation script; scripts are named with a number prefix, for example, `5_autoscaling.sh` or `9_agent.sh`, and run in reverse numeric order.

{name}/ is a private directory for the nodepart.

setup/ is intended for one-time setup of the nodepart. The script, `setup/setup.py`, is started by the activation script with the associated parameters specified in the topology document. The setup/ directory is deleted after the setup/setup.py script returns.

• Setup script

If present, the setup/setup.py script is started with the associated parameters. For example, the workload agent nodepart is configurable for the installation location, IaaS server and a command port; parameters are specified in the topology document within the node-part object, such as this excerpt from the sample topology document in the section, Application model and topology document examples:

```
"node-parts": [  
  {  
    "parms": {  
      "iaas-port": "8080",  
      "agent-dir": "/opt/IBM/maestro/agent",  
      "http-port": 9999,  
      "iaas-ip": "127.0.0.1"  
    },  
    "node-part": "https://localhost:9444\storehouse\admin\plugins\agent\nodeparts\agent-linux-x64.tgz"  
  },  
  {  
    "parms": {  
      "iaas-port": "8080",  
      "agent-dir": "/opt/IBM/maestro/agent",  
      "http-port": 9999,  
      "iaas-ip": "127.0.0.1"  
    },  
    "node-part": "https://localhost:9444\storehouse\admin\plugins\agent\nodeparts\agent-linux-x64.tgz"  
  },  
  {  
    "parms": {  
      "iaas-port": "8080",  
      "agent-dir": "/opt/IBM/maestro/agent",  
      "http-port": 9999,  
      "iaas-ip": "127.0.0.1"  
    },  
    "node-part": "https://localhost:9444\storehouse\admin\plugins\agent\nodeparts\agent-linux-x64.tgz"  
  }
```

The setup/setup.py script can import the maestro module. Configuration parameters are available as maestro.parms, for example:

```python
import json
import os
import subprocess
import sys
import maestro
```
parms = maestro.parms

subprocess.call('chmod +x *.sh', shell=True)

rc = subprocess.call('./setup_agent.sh "%s" %d %s %s' % (parms['agent-dir'],
                            parms['http-port'], parms['iaas-ip'], parms['iaas-port']), shell=True)
maestro.check_status(rc, 'setup_agent.sh: rc == %s' % rc)

The implementation is arbitrary. The previous example passes the parameter values to a shell script, which in turn starts sed for token replacement in the support scripts.

Nodeparts are generally added to vm-templates during the resolve phase of deployment. There are two ways that nodeparts are added to a vm-template:

– Explicit inclusion as part of a named package.
– Implicit inclusion as part of a matching default package.

For example, the config.json file from the workload agent plug-in is shown here:

```json
{
    "name":"agent",
    "packages":{
    "default":[
    {
    "requires":{
    "arch":"x86_64",
    "os":{
    "RHEL":"*"
    }
    },
    "node-parts":[
    {
    "node-part":"nodeparts/agent-linux-x64.tgz",
    "parms":{
    "agent-dir":"/opt/IBM/maestro/agent",
    "http-port":9999,
    "iaas-ip":"@IAAS_IP@",
    "iaas-port":"@IAAS_PORT@"
    }
    }
    ]
    }
    },
    "node-parts":[
    {
    "node-part":"nodeparts/agent-linux-x64.tgz",
    "parms":{
    "agent-dir":"/opt/IBM/maestro/agent",
    "http-port":9999,
    "iaas-ip":"@IAAS_IP@",
    "iaas-port":"@IAAS_PORT@"
    }
    }
    ]
}
```

In general, config.json may define any number of named packages. The previous example shows one package named “default”. Each package is an array containing any number of objects, where each object is a candidate combination of node-parts and/or parts. The candidates are specified by mapped values for requires, node-parts and parts. Package contents are additive within a pattern type. For example, if two plug-ins are part of the same pattern type and both define package “FOO” in config.json, then resolving package “FOO” considers the union of candidates from both config.json files.

“default” is a special package name. The resolve phase always includes the default package; other named packages are resolved only when explicitly named.

Requires
The value of requires is an object that qualifies the candidate in terms of applicable operating system and architecture. Supported keys and mapped values are as follows:

Key arch; mapped value is a string or array of strings. Supported strings include x86 and x86_64 (ESX).

For example,

"arch" : ["x86", "x86_64"]

Key os; mapped value is an object of os name/version pairs. Supported os name strings include RHEL and AIX. Valid expressions include the following:

- "*" – matches all versions
- "[a, b]" – matches all versions between a and b, inclusive
- "(a, b)" – matches all versions between a and b, exclusive
- "[* , b]" – matches all versions up to b, inclusive
- "[a, *)" – matches all versions a and greater

Package names must be unique across all installed plug-ins, except for the reserved default package name. The default package is an additive. All plug-ins can append candidates to the default package and all candidates are automatically evaluated for a match during the resolve phase of deployment. Thus, the default package is the basis for implicit inclusion; all other named packages are only included explicitly.

- Standard nodeparts

All vm-templates include workload agent and firewall nodeparts. Each of the nodeparts defines a standard interface for integration with and used by other nodeparts.

- Workload agent

  The workload agent is an OSGi-based application responsible for installing parts and driving the life cycle of the roles and dependencies in a plug-in.

  The workload agent processes the following sequence of operations to drive roles and dependencies toward a running application:

  1. Get the virtual machine vm-template from the topology document, for example, if SERVER_NAME == application-was.1, then get the vm-template named application-was.

  2. For each part in the vm-template, run the following sequentially:

     a. Download the part .tgz file and extract the file into {tmpdir}.

     b. Start {tmpdir}/install.py, passing any associated parameters specified in the topology document.

     c. Delete {tmpdir}.

  3. For each role in the vm-template run the following concurrently:

     a. Start {role}/install.py, if exists.

     b. For each dependency of the role, start {role}/{dependency}/install.py, if exists.

     c. Start {role}/configure.py, if exists.

     d. For each dependency of the role, start {role}/{dependency}/configure.py, if exists.

     e. Start {role}/start.py, if exists.

  4. React to the changes in the dependencies ({role}/{dependency}/changed.py) and peers ({role}/changed.py), if exists. Role status is automatically advanced up to CONFIGURING, but not to RUNNING. A plug-in
script must set the role status to **RUNNING**. Role status is set only by the `{role}/start.py` and `changed.py` scripts (role or dependency). Role status is set as follows:

```python
import maestro
maestro.role_status = 'RUNNING'
```

There are several custom features of the workload agent, including the following:

- The workload agent is extensible.
- Other nodeparts can install features into the OSGi-based application.

Complete the following steps to install nodepart features into the agent:

1. Provide a `.tgz` file containing the following files:
   - `lib/{name}.jar` - bundle Java archive (JAR) files
   - `lib/features/featureset_{name}.blst` - list of bundles for the feature set
   - `usr/configuration/{name}.cfg` - OSGi configuration code
2. Provide a start script before slot 9 that installs the `.tgz` file contents into the agent.

Other nodeparts do not need to know the installation location of the agent application. Rather, the agent provides the shared script, `agent_install_ext.sh`, to install custom features. Shared scripts are always on the PATH, so a typical start script for a nodepart to install a custom feature is as follows:

```sh
#!/bin/sh
agent_install_ext.sh ../../autoscaling/autoscaling.tgz
```

The `agent_install_ext.sh` script extracts the contents of the specified `.tgz` file into the agent application. The custom feature is included when the agent is started.

- **Firewall**

  The firewall nodepart defines a generic API for shell scripts and Python scripts to manipulate the firewall settings on the virtual machine. The default implementation for RHEL is based on iptables; other implementations can be built. The `firewall.sh` script is the shell script provided for Linux. The following content summarizes the shell usage:

  ```sh
  firewall.sh open tcpin [-src <src>] [-sport <port>] [-dest <dest>] [-dport <port>]
  firewall.sh open tcpout [-src <src>] [-sport <port>] [-dest <dest>] [-dport <port>]
  ```

  ```sh
  firewall.sh close tcpin [-src <src>] [-sport <port>] [-dest <dest>] [-dport <port>]
  firewall.sh close tcpout [-src <src>] [-sport <port>] [-dest <dest>] [-dport <port>]
  ```

  The `open tcpin` directive is tailored for TCP connections and opens corresponding rules in the INPUT and OUTPUT tables to allow request and response connections. The `open in` directive opens the INPUT table only. For `src` and `dest`, `private` is a valid value. This value indicates that `<src>` and
<dest> are limited to the IP range defined for the cloud. The value private is defined in the config.json file for the firewall plug-in as follows:

```
{
  "name":"firewall",
  "packages":{
    "default":{
      "requires":{
        "arch":"x86_64",
        "os":{
          "RHEL":"*"
        }
      },
      "node-parts":[
        {
          "node-part":"nodeparts/firewall.tgz",
          "parms":{
            "private":"PRIVATE_MASK"
          }
        }
      ]
    }
  }
}
```

Currently, the value of PRIVATE_MASK is set as part of the maestro provisioning steps. The cloud-specific value is found in the cloud project build.xml file, for example, cloud.HSLT/build.xml. For an Orion cloud, PRIVATE_MASK == 10.0.0.0/255.0.0.0.

The Python API is similar. Callers must import the maestro package. The provided firewall methods are:

- `maestro.firewall.open_in(**args)`
- `maestro.firewall.open_out(**args)`
- `maestro.firewall.open_tcpin(**args)`
- `maestro.firewall.open_tcpout(**args)`
- `maestro.firewall.close_in(**args)`
- `maestro.firewall.close_out(**args)`
- `maestro.firewall.close_tcpin(**args)`
- `maestro.firewall.close_tcpout(**args)`

where **args represents keyword args.

Valid keywords include: protocol, src, sport, dest, and dport.

The following is an example of one valid invocation:
```
maestro.firewall.open_tcpin(src='private', dport=8080)
```

**Parts**

Parts are installed by the workload agent and generally contain binary and lifecycle scripts associated with roles and dependencies. Review the following information about parts:

- **Conventions**
  All parts must have an install.py script at the root. Additional files are allowed.

- **Common scripts**
  By default, the maestro package contains the following functions:
- **maestro.download(url, f)**: Downloads the resource from the url and saves the resource locally as file f.
- **maestro.downloadx(url, d)**: Downloads and extracts a .zip, .tgz, or .tar.gz file into directory d. The .tgz and .tar.gz files are streamed through extraction; the .zip file is downloaded and then extracted.
- **maestro.decode(s)**: Decodes strings encoded with the maestro encoding utility, such as from a transformer using com.ibm.ws.security.utils.XOREncoder.encode(String).
- **maestro.install_scripts(d1)**: Utility function for copying life cycle scripts into {scriptdir} and making the shell scripts executable (dos2unix and chmod +x).
- **maestro.check_status(rc, message)**: Utility function for logging and exiting a script for non-zero rc.

**Data objects**

The agent appends data objects or dictionaries to the maestro package when starting part installation scripts as follows:

- **maestro.parturl**: fully-qualified URL from which the part .tgz file was obtained (string; RO)
- **maestro.filesurl**: fully-qualified URL prefix for the shared files in storehouse (string; RO)
- **maestro.parms**: associated parameters specified in the topology document (JSON object; RO)
- **maestro.node['java']**: absolute path to Java executable (string; RO)
- **maestro.node['deployment.id']**: deployment ID, for example, d-xxx (string; RO)
- **maestro.node['tmpdir']**: absolute path to working directory. This path is cleared after use (string; RO)
- **maestro.node['scriptdir']**: absolute path to the root of the script directory (string; RO)
- **maestro.node['name']**: server name (same as env variable SERVER_NAME) (string; RO)
- **maestro.node['instance']['private-ip']** (string; RO)
- **maestro.node['instance']['public-ip']** (string; RO)
- **maestro.node['parts']**: shared with all Python scripts invoked on this node (JSON object; RW)

**Roles**

A role represents a managed entity within a virtual application instance. Each role is described in a topology document by a JSON object, which is contained within a corresponding vm-template like the following:

```
maestro.role['tmpdir'] : role-specific working directory; not cleared (string; RO)
```

You can import custom scripts, for example, import my_role/my_lib.py:

```python
utilpath = maestro.node['scriptdir'] + '/my_role'
if not utilpath in sys.path:
sys.path.append(utilpath)
import my_lib
```

The following is an example role from a topology document:

```json
"roles":[
{
  "plugin":"was/2.0.0.0",
  "parms":{
    "ARCHIVE":"$$1",
    "USERID":"virtuser",
    "PASSWORD":"$$6"
  },
  "depends":[
  {
    "role":"database-db2.DB2",
    "parms":{
      "db_provider":"DB2 Universal JDBC Driver Provider",
      "jndiName":"TradeDataSource",
      "inst_id":1,
      "POOLTIMEOUT":"$$11",
```
"NONTRAN":false,"db2jarInstallDir":"\opt\db2jar","db_type":"DB2","db_dsname":"db2ds1","resourceRefs":[
    {
        "moduleName":"tradelite.war",
        "resRefName":"jdbc\TradeDataSource"
    }
],
"db_alias":"db21"
],
"type":"DB2",
"bindingType":"javax.sql.DataSource"}

Role state and status

The agent implements a state machine that drives each role through a basic progression as follows:

INITIAL > INSTALLED > CONFIGURING > STARTING > RUNNING

The role status can change during transitions and within a state. Here is the same state progression, shown with the details of status and life cycle scripts started:

**Table 72. Role state and status**

<table>
<thead>
<tr>
<th>Role state script</th>
<th>Transition</th>
<th>Update status</th>
<th>Set role status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>Initial =&gt; INSTALLED</td>
<td>on entry</td>
<td>INITIAL</td>
</tr>
<tr>
<td>INSTALLED</td>
<td>INSTALLED =&gt; RUNNING</td>
<td>during</td>
<td>INSTALLING CONFIGURING STARTING (role status by script)</td>
</tr>
<tr>
<td>{role}/install.py then all {role}/{dep}/install.py</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{role}/configure.py then all {role}/{dep}/configure.py</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{role}/start.py</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RUNNING</td>
<td>on entry on changed</td>
<td>role_status (set by script)</td>
<td></td>
</tr>
<tr>
<td>{role}/start.py</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the process is stopped or an unrecoverable error occurs, move the role to a TERMINATED or ERROR state. If an error is recoverable, you can keep the role state as RUNNING and change the role status to FAILED.. For example, if WebSphere Application Server crashes, and the crash is detected by wasStatus.py the wasStatus.py script sets maestro.role_status = "FAILED". When a user starts WebSphere Application Server from the Virtual Application Console, one of the following processes occurs:

- If there is no dependency, operation.py sets maestro.role_status="RUNNING".
- If WebSphere Application Server depends on another role (such as DB2), operation.py sets maestro.role_status="CONFIGURING". The was/{dep}/changed.py script starts as the result of role status changing from FAILED to CONFIGURING, and the script starts WebSphere Application Server, processes dependency info from maestro.deps and sets maestro.role_status="RUNNING".
Two status checks are available that determine when a dependency script is started on a source role. For example, if A depends on B, then A is the source and B is the target as follows:

- Role A must be in the `RUNNING` state (Role.rolesChanged())
- Role B must have status `== RUNNING` (Relation.rolesChanged())

**Existing resources**

Plug-ins can interact with existing resources. Although the existing resource is not a managed entity within the plug-in, it is modeled as a role. This allows for a consistent approach, whether dealing with pattern-deployed or existing resources. Specifically:

- Integration between resources is modeled as a dependency between two roles. The target role (pattern-deployed or existing) exports properties that are used by a dependency script on the source `{{role}}/{{dep}}/changed.py` to realize the integration. This design provides reuse of the source dependency script. For example, in the wasdb2 plug-in, the `WAS/DB2/changed.py` script manages a WebSphere Application Server data source for any pattern-deployed or existing database.

- User interactions in the IBM Cloud Orchestrator deployment user interface are consistent for resources and integrations. Resources (pattern-deployed or existing) are represented as roles, meaning they display on the **Operations** tab of the deployment panel in the product user interface. For example, you can look for a role when you change a password. For a pattern-deployed resource, the change is applied to the resource, then exported for dependencies to react. For an existing resource, change is exported for dependencies to react like when the password is already changed externally.

Managing configuration of the interactions (links) is handled through the source role.

An existing resource is modeled by a component in `appmodel/metadata.json` file. Typical component attributes are required to connect to the resource, such as hostname/IP address, port and application credentials.

Integration with existing resources is modeled by a link in the `appmodel/metadata.json` file.

If a type of resource displays as pattern-deployed or existing, then consolidation is possible by adding a role to represent the external resource. This role can export parameters from the existing resource that the dependent role for the pattern deployed case can handle.

Consider the case of an application using an existing resource, such as wasdb2, imsd3 and wasctg plug-ins. At the application model level, the existing database is a component, and WebSphere Application Server uses it, on behalf of the application, as a represented link to that component. Typical attributes of the existing database are its host name or IP address and port, and the user ID and password for access.

In older service approaches, the existing database component has a transform that builds a JSON target fragment that stores the attributes, and the link transform uses these attributes. In IMS, for example, the link transform creates a dependency in the WebSphere Application Server role in the WebSphere Application Server node, with the parameters of the existing database passed from the component. The dependent role `configure.py` script is used to configure WebSphere.
Application Server to use the existing database based on the parameters. This is sufficient, but in the deployment panel, the parameters of the existing database appear in the WebSphere Application Server role, which is not sensible.

In the new role approach, the target component creates a role JSON object and the link transform adds it to the WebSphere Application Server virtual machine template list of roles. The wasdb2 plug-in creates an xDB role to connect to existing DB2 and Informix databases. IMS can convert to this model, and move its configure.py and change.py scripts to a new xIMS role. The advantage here is in the deployment panel, which lists each role for a node separately in a left column where its parameters and operations, are better separated for user access.

The wasdb2 plug-in provides an additional feature that IMS and CTG might not use. The plug-in also supports pattern-deployed DB2 instances. In the pattern-deployed scenario, the DB2 target node is a node that is started. The correct model is a dependent role and the link configuration occurs when both components, source WebSphere Application Server and target DB2, start. The changed.py script is then run. For the existing database scenario, the wasdb2 plug-in exports the same parameters as the DB2 plug-in, and then processing for pattern-deployed and existing cases can be performed in the changed.py script.

IMS and wasctg do not require this process and can use a configure.py role script for new roles.

Set repeatable task

At run time, a role might need to perform some actions repeatedly. For example, the logging service must back up local logs to the remote server in a fixed period. The plug-in framework allows a script that is started after a specified time, to meet this requirement.

In any part script, such as configure.py and start.py, you can specify a task as follows:

```python
task = {}
task['script'] = 'backupLog.py'
task['interval'] = 10

taskParms={}
taskParms['hostname'] = hostname
taskParms['directory'] = directory
taskParms['user'] = user
taskParms['keyFile'] = keyFile
task['parms'] = taskParms
maestro.tasks.append(task)
```

You must have a dictionary object named task. You can change this to another valid name. The target script is specified by task['script'] and the interval is specified by task['interval']. You can add parameters to the script by using task['parms']. This is an optional addition to the script. The, maestro.tasks.append(task) is used to enable this task. In this sample, backupLog.py, which is located in the folder {role}/scripts, is started after 10 seconds when the current script is completed. Using the backupLog.py script, you can retrieve the task parameters from maestro.task['parms'] and retrieve the internal from maestro.task['interval']. This script is only started one time. If the backupLog.py script is required to be started repeatedly, you must add the same codes into the backupLog.py script. When the current script is completed, it is started after the new specified internal and parameters.
Recovery: Reboot or replace?

If a virtual machine stops unexpectedly, the master agent recovers the failed virtual machine. The action depends on the virtual machine type. A persistent virtual machine is rebooted. Other virtual machines are replaced.

A virtual machine is persistent if it is an instance of a vm-template with a true-valued persistent property, as follows:

```
"vm-templates": [
  {
    "persistent": true,
    "scaling": {
      "min": 1,
      "max": 1
    },
    "name": "Caching_Master",
    "roles": [
      {
        "depends": [{
          "role": "Caching_Slave.Caching"
        }],
        "type": "CachingMaster",
        "name": "Caching",
        "parms": {
          "PASSWORD": "$XSAPassword"
        }
      }
    ],
    "packages": ["CACHING"
      ],
  }
],
```

There are two ways for a plug-in to mark a virtual machine as persistent:

- Direct
  The transformer adds the persistent property to the vm-template.
- Indirect
  The package configuration specifies the persistent attribute.

The direct method supersedes the indirect. That is, if the vm-template is marked persistent (true or false), that is the final value. If the vm-template is not marked persistent, the resolve phase of deployment derives a persistent value for the vm-template based on the packages associated with that vm-template. The vm-template is marked persistent or true if any package declares persistent true.

The indirect method provides more flexibility to integrate parts and node parts, without requiring global knowledge of where persistence is required. A transformer adds the property as follows:

```
"vm-templates": [
  {
    "persistent": true,
    "scaling": {
      "min": 1,
      "max": 1
    },
    "name": "Caching_Master",
    "roles": [
      {
        "depends": [{
          "role": "Caching_Slave.Caching"
        }],
        "type": "CachingMaster",
        "name": "Caching",
        "parms": {
          "PASSWORD": "$XSAPassword"
        }
      }
    ],
    "packages": ["CACHING"
      ],
  }
],
```

Package configuration is specified in the config.json file as follows:

```
{
  "name": "db2",
  "version": "1.0.0.0",
```
"packages" : {
  "DB2" : [{
    "requires" : {
      "arch" : "x86_64",
      "memory" : 0,
      "persistent" : true,
    },
    "parts" : [
      {
        "part" : "parts/db2-9.7.0.3.tgz",
        'parms' : {
          "installDir" : "/opt/ibm/db2/V9.7"},
        "part" : "parts/db2.scripts.tgz"
      }
    ]
  }
}

Virtual Application Console

You can view the virtual application instances through the Operations tab of the Virtual Application Console.

Operation

- Define an operation.

Create a JSON file named, operation.json, in the plugin/appmodel directory. Each operation.json file should contain a JSONObject. The following code example shows a part of the WebSphere Application Server operation.json file, where the key is the role type WAS, and the value is a JSONArray that contains these operation definitions:

```
"WAS": [
  {
    "id": "setWASTrace",
    "label": "TRACE_LEVEL_LABEL",
    "description": "TRACE_LEVEL_DESCRIPTION",
    "script": "debug.py setWASTrace",
    "attributes": [
      {
        "label": "TRACE_STRING_LABEL",
        "type": "string",
        "id": "traceString",
        "description": "TRACE_STRING_DESCRIPTION"
      }
    ]
  }
]```

where

- script defines the operation debug.py script that is started when the operation is submitted. The operation script name can also be followed by a method name such as setWASTrace that is included in the previous code sample. The method name can be retrieved later in the operation script. The operation script should be placed under the role scripts path, for example, plugin/parts/was.scripts/scripts/WAS.
- attributes define the operation parameters that you must input. The operation parameters can be retrieved later by the operation script.

- Attributes for operations against multiple instances.

If a role has more than one instance, you can use these attributes in the operation definition to control how an operation is applied to instances. The following attributes are validated if a role has more than one instance:
### rolling
Determine if an operation is performed sequentially or concurrently on instances.
- To perform an operation concurrently, set "rolling": false. This is the default setting.
- To perform an operation sequentially, set "rolling": true

### target
Determines if an operation is performed on a single instance or all instances.
- To perform an operation on all instances, set "target": All. This is the default setting.
- To perform an operation a single instance, set "target": Single

See the WebSphere Application Server operation.json file for an example.

#### Setting a particular role status for an operation.
By default, when an operation is being performed, the role status is set to "CONFIGURING" and then is set back to "RUNNING" when the operation is complete. This change in status can sometimes stop the application itself. Some operations, such as exporting logs, do not require a role to change its status from "RUNNING". For these types of operations, you can explicitly set the role status to use when the operation starts. For example, to keep the role status as "RUNNING" when the operation starts, add the following attribute to the operation definition:

"preset_status": "RUNNING"

The role status will remain as "RUNNING" unless an error occurs during the operation.

See the WebSphere Application Server operation.json file for an example.

#### Operation script
The operation script can import the maestro module. The information retrieved in the role life cycle part script can be retrieved the same way in the deployment panel, such as maestro.role, maestro.node, maestro.parms, and maestro.xparms. Also, all of the utility methods, such as download and downloadx, can be used. Parameters that are configured at the deployment panel are passed into the script and are retrieved at maestro.operation['parms']. The method name defined in the operation.json file is retrieved at maestro.operation['method'], and operation ID is retrieved at maestro.operation['id'].

#### File download and upload
All downloaded artifacts must be placed under the fixed root path. The operation script can get the root path from maestro.operation['artifacts_path']. To specify a file downloaded later, insert maestro.return_value="file://key.p12" in the script. The prefix, file://, indicates that a file is required for download. After the script is complete, the deployment panel displays a link to download the file. Uploaded files are placed in a temporary folder under the deployment path in the storehouse. The operation script retrieves the full storehouse path of the uploaded files, for example:

uploaded_file_path=maestro.operation['parms']['parm_name']

After the file path is retrieved, the maestro.download() method downloads the file. When the operation is complete, the temporary files in storehouse are deleted. When the operation script interacts with kernel services and storehouse, the script prefers to use the authorization token passed from the user interface, but not use the agent token, so that the operations can be audited later. The
operation script should use maestro.operation("user_token") to retrieve the user
token passed from the user interface, which contains user uuID. You can use it
later in communication with kernel services and storehouse. For example, to
upload a file into storehouse, use upload_if_new(url, file) which uses
agent_token to generate the authorization header. You can also use
upload_if_new(url, file, user_token), where the passed-in user_token is used to
generate the header.

- **Operation result**
  
  You can specify the result of an operation using
  maestro.operation['successful'] = True/False. The default value is True. After the
  operation script completes successfully, the user interface displays the result as
  SUCCESS or FAIL. If the script ran with failures such as return code! = 0, the user
  interface result displays as ERROR, and the responding role changes to ERROR
  state. If you want a more meaningful result returned, insert
  maestro.operation['return_value'] = "my return value". The return value displays
  on the user interface when the script completes.

- **Depends-on role operation**

  In some scenarios, one role update causes another role to also need an update.
  This is called the depends-on role operation. An example of this is if DB2
  changes a password, WebSphere Application Server also updates the data source
  configuration. In the operation script, export the changed value, such as:
  maestro.export['DB_PASSWORD'] = "<XOR>UIK8CNz". Then, change the
depends-on role changed.py script, to add code to handle the other update.

```python
if len(updated) == 1:
    myrole = updated[0]
    if deps[myrole]['is_export_changed']:
        print 'export data changed'
    else:
        print 'only status changed'
```

**Configuration**

In the deployment panel, a configuration update is handled as a special type of
operation. The configuration processes include:

- **Define a configuration.**

  Add the tweak.json file under the plug-in plugin/appmodel folder, to specify
  which configuration parameters can be changed during run time. This means
  that some parameters in the topology model can be changed and validated at
  run time. Each tweak.json file is a JSONArray. Each object describes a parameter
  that can be tweaked in the topology model. For example, in the WebSphere
  Application Server plug-in, add the following code example to a tweak.json file.
  The parameter "ARCHIVE" under the "WAS" role can be tweaked.

  The value of "id" is composed with {role_type}.{parm_name}. Other attributes
  such as "label" and "description" are prepared for the user interface. This is
  similar to the definition in the metadata.json file located in the /appmodel
directory.

  ```json
  {
    "id": "WAS.ARCHIVE",
    "label": "WAR/EAR File",
    "description": "Specifies the web/enterprise application to be uploaded. ",
    "type": "file",
    "extensions": [
      "war",
      "ear"
    ]
  }
  ```
For a parameter under the depends section, the value of "id" is composed with 
{role_type}.{depends_role_type}.{parm_name}:

```
{
  "id":  "WAS.DB2.MINPOOLSIZE",
  "type":  "number"
}
```

To enable this feature, you must add the following code to the operation.json file:

```
{
  "id":  "configuration",
  "label":  "CONFIGURATION_LABEL",
  "description":  "CONFIGURATION_DESCRIPTION",
  "script":  "change.py"
},
```

- **Configuration script: change.py**
  The `change.py` script is similar to the operation script, except for the following:
  - When you use the depends role configuration, the `change.py` script is placed under the plugin/parts/{parts_package}/scripts/{role_type}/
    {depends_role_type} path.
  - After the script is started successfully, the changed values are automatically persisted to storehouse.
  - For the artifacts-type configuration update, such as when you update the WebSphere Application Server ARCHIVE, if the `change.py` script completes successfully, the artifacts in temp folder in storehouse are cloned to the /deployment/{deployment_id}/artifacts path before they are deleted.

- **Configuration for non-role component, such as remote DB2 and Tivoli Directory Service.**
  If the transformer needs configuring, use wasxdb2 as shown in the example:
  In XDB2Transformer.java:
  ```java
  Change
  result.put("service", attributes);
  ```
  to
  ```java
  result.put("attributes", attributes);
  result.put("service", prefix);
  ```
  In WASXDB2Transformer.java:
  ```java
  Change
  JSONObject serviceParms = (JSONObject) targetFragment.get("service");
  To
  JSONObject serviceParms = (JSONObject) targetFragment.get("attributes");
  ```
  and add following line:
  ```java
  //WASxDB2 acts as an extension, not a dependency (so, no role defined)
  depend.put("type", "xDB2");
  depend.put("service", targetFragment.get (service));
  ```

- **Configuration for multi-links between two components, such as remote WebSphere Application Server and Tivoli Directory Service.**
  Besides the tweak.json file, the transformer needs specific codes to handle this scenario. The target topology segment should look like the following:
  ```json
  "depends": [ 
    { "role": "User_Registry-tds.TDS1",
```
The "parms" are nested structure and you must specify a "deps_key_id" as the key for the subgroup in the "parms". You can use Java based code or a template to complete the transformer. In parts scripts changed.py and change.py, you can retrieve the parameters using "for cycle" as follows:

```python
for key in parms:
    roleParms = parms[key]
    print key
    print roleParms['xLDAP_ROLE_USER_MAPPING']
    print roleParms['xLDAP_ROLE_GROUP_MAPPING']
    print roleParms['xLDAP_SPECIAL_SUBJECTS_MAPPING']
```

Other reference

See the related links for references to JSON formatting and validation, and, guidelines for starting external commands from Python scripts.

Application model and topology document examples

The application model and topology documents are core pieces of the IBM Cloud Orchestrator modeling and deployment. This section presents examples of these related documents as a basis for the other sections in this guide. The sample Java Enterprise Edition (Java EE) web application provided with the web application virtual application pattern type is used as an example.

Application model

The appmodel.json file represents the serialization of the model that is defined in the Pattern Builder user interface. Components (nodes) and links, along with user-specified property values, are represented.

```json
{
    "model":{
        "name":"Sample",
        "nodes":{
            "attributes":{
                "WAS_Version":"7.0",
                "archive":"artifacts/tradelite.ear",
                "clientInactivityTimeout":60,
                "asyncResponseTimeout":120,
                "propogatedOrBMTTranLifetimeTimeout":300,
                "totalTranLifetimeTimeout":120
            },
            "id":"application",
            "type":"EAR"
        }
    }
}
```
Topology document

The final topology document for a given application model depends on the deployment environment, such as storehouse URL and image ID. This sample shows two vm-templates from the web application, application-was and database-db2. Each vm-template has a list of nodeparts and parts to be installed, and run time roles to be managed.

```json
{
"vm-templates": [{
"parts": [
{
"part": "https:\/\/localhost:9444\/storehouse\/admin\/plugins\/was\/parts\/was-7.0.0.11.tgz",
"parms": {
"installDir": "/opt"
}
},
{
"part": "https:\/\/localhost:9444\/storehouse\/admin\/plugins\/was\/parts\/was.scripts.tgz"
},
{
"part": "https:\/\/localhost:9444\/storehouse\/admin\/plugins\/wasdb2\/parts\/db2.jdbc.tgz",
"parms": {
"installDir": "/opt/db2.jar"
}
},
{
"part": "https:\/\/localhost:9444\/storehouse\/admin\/plugins\/wasdb2\/parts\/wasdb2.scripts.tgz"
}
],
"node-parts": [
{
"parms": {
"private": "127.0.0.1"
}
},
"node-part": "https:\/\/localhost:9444\/storehouse\/admin\/plugins\/firewall\/nodeparts\/firewall.tgz"
},
{
"parms": {
"iaas-port": "8080",
"agent-dir": "/opt/IBM/maestro/agent",
"http-port": "9990",
"jndi-data-source": "jdbc/TradeDataSource",
"xa-data-source": false,
"max-connection-pool": 10,
"min-connection-pool": 1,
"nontransactional": false,
"connection-timeout": 180
}
],
"id": "WASDB2",
"type": "WASDB2",
"source": "application",
"target": "database",
"annotation": "",
"attributes": {
"connection-timeout": 180,
"nontransactional": false,
"min-connection-pool": 1,
"jndi-data-source": "jdbc/TradeDataSource",
"xa-data-source": false,
"max-connection-pool": 10
}
]
}
```
"iaas-ip": "127.0.0.1",
"node-part": "https://localhost:9444/storehouse/admin/plugins/agent/nodeparts/agent-linux-x64.tgz"
},

'parms': {
  "installerURL": "files\itmosv6.2.Zfp2_linuxx64.tar.gz",
  "omnibusTarget": "",
  "temsiip": "",
  "omnibusip": ""
},

"node-part": "https://localhost:9444/storehouse/admin/plugins/monitoring/nodeparts/monitoring.tgz"
},

"node-part": "https://localhost:9444/storehouse/admin/plugins/deployinlet/nodeparts/deployinlet.tgz"
},

'parms': {
  "collectors": [
    {
      "url": "http://COLLECTOR_NODE_IP:8080"
    }
  ],

  "node-part": "https://localhost:9444/storehouse/admin/plugins/logging/nodeparts/logging.tgz"
},

"node-part": "https://localhost:9444/storehouse/admin/plugins/cloud.HSLT/nodeparts/iaas.tgz"
},

"node-part": "https://localhost:9444/storehouse/admin/plugins/autoscaling/nodeparts/autoscaling.tgz"
},
"scaling": {
  "min": 1,
  "max": 1
},

"image": {
  "type": "medium",
  "image-id": "none",
  "activators": [
    "https://localhost:9444/storehouse/admin/clouds/mocked2.zip"
  ],

  "name": "application-was",
  "roles": [
    {
      "depends": [
        {
          "role": "database-db2.DB2",
          "parms": {
            "MAXPOOLSIZE": "$2",
            "installDir": "\opt\db2jar",
            "inst_id": 1,
            "POOLTIMEOUT": 180,
            "NONTRAN": false,
            "DS_JNDI": "jdbc\TradeDataSource",
            "MINPOOLSIZE": "$3"
          },
          "type": "DB2"
        }
      ],
      "parms": {
        "clientInactivityTimeout": "60",
        "ARCHIVE": "$1!",
        "propogatedOrBMTTranLifetimeTimeout": "300",
        "asyncResponseTimeout": "120",
        "USERID": "virtuser",
        "totalTranLifetimeTimeout": "120",
        "PASSWD": "xor-BW4SbZM9FhwuFgUxEy0YzEYyQ44="
      },
      "external-uri": "http://\{SERVER\}:9080/",
      "type": "WAS",
      "name": "WAS",
      "requires": {
        "memory": 256
      }
    }
  ]
}
"name":"database-db2",
"roles":[
{
  'parms':{
    "DB_PORT":50000,
    "DB_PATH":"/home/db2inst1",
    "PASSWD":"<xor>NRA0aWgHOGoRaG47DiU=",
    "DB_NAME":"adb",
    "SQL_URL":
    "https://localhost:9444/storehouse/user/applications\ /
    a-6d1ac0d4-4e4c-49d7-954f-d4884a6ad703/artifacts\/setup_db.sql"
  },
  'external-uri':"jdbc:db2://{SERVER}:50000/adb;user=db2inst1;password=jOk67Xg5N71dQz;",
  'type':'DB2",
  'name':'DB2'
}
],
"packages":[
  "DB2"
]
}

Related information:

- JSON Formatter and Validator
- JSONLint: The JSON Validator
- Python documentation: Subprocess management

Plug-ins for development:

The Plug-in Development Kit (PDK) includes several plug-ins that you can install to assist you with testing and troubleshooting your plug-ins.

Debug:

The debug component, com.ibm.maestro.plugin.debug, provides support for developing and debugging plug-ins.

Before you begin

The debug component is included in the IBM Plug-in Development Kit.

Before you can use the debug plug-in, you must import it into your IBM Cloud Orchestrator development environment.

1. Import the com.ibm.maestro.plugin.debug plug-in.
2. Restart IBM Cloud Orchestrator.

About this task

The following are the attributes for the debug component:

- Do not deploy this application model, only write topology document to storehouse: Select this option to specify the use of the Storehouse Browser to get a topology document.

This option enables a deploy-only mode so that your virtual application pattern is transformed through the complete deployment process, but no virtual machines are created. Rather, the finalized topology document is written to the Storehouse. You can use the Storehouse Browser to view your final topology document.
Using this option is the first debugging step after writing the plug-in and pattern type. To locate the Storehouse Browser, access the product user interface. From the menu, click **System > Storehouse Browser.** Expand **User Deployments > deployment ID.** Click **topology.json > Get Contents.**

The deployment ID is in the Kernel Services (KS) console log, as follows:

```
```

**• Leave files on deployed nodes to enable manual debugging:** Select this option if you want to use Secure Shell (SSH) to access the node for manual debugging.

The virtual application pattern is deployed when this check box is selected. Virtual machines are created in the deployment process.

The files are retained on deployed virtual machines so that you can log in to the machines with SSH. You can view files and re-run nodepart and part life-cycle scripts. This option supports greater productivity in debugging these scripts, because you can edit and re-run scripts on the node and are not required to fully deploy new nodes to start a test cycle. See Running Scripts on deployed virtual machines for instructions on using this option.

**Procedure**

You can view, edit, or add this virtual application component in the user interface as follows:

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns.**
2. Select a **virtual_application_pattern.**
3. Click **Edit the virtual application** icon located in the upper right corner of the Pattern Builder palette.
4. To add the debug component to a virtual application pattern, click **Debug** listed under **Other Components** and drag the icon to the Pattern Builder canvas. The properties panel for the database component displays to the right of the Pattern Builder palette. For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
5. To edit an existing debug component, select the **Debug** part on the Pattern Builder canvas. The properties panel displays. For more details on the properties panel settings, see the properties descriptions or view the help by selecting the help icon on the properties panel.

For more details on the properties panel settings, view the help by selecting the help icon on the properties panel.
6. You can also view the debug component properties by viewing the plug-in information.

   Click **PATTERNS > Deployer Configuration > System Plug-ins.** Select **Foundation Pattern Type** from the **Select a pattern type** menu.

   The plug-ins included with the Foundation Pattern are listed. Select **com.ibm.maestro.plugin.debug/x.x.x.x** from the **System Plug-ins** palette where **x.x.x.x** corresponds to the version numbers. The component plug-in configuration information displays on the canvas.

**Results**

You have edited a current component, edited an existing component, or added one.
Unlock:

To facilitate plug-in development, the unlock plug-in provides the ability to delete a plug-in used by a virtual application instance so that you can easily replace the plug-in with an updated version and activate the new plug-in on existing virtual machines instead of redeploying a new copy of the application.

Before you begin

Important: The unlock plug-in is for development environments only to facilitate testing of plug-ins that are being developed. It is not intended for production environments and should not be installed in a production environment.

About this task

The debug component is included in the IBM Plug-in Development Kit.

In a normal IBM Cloud Orchestrator environment, a plug-in cannot be removed if it is being used by a deployed virtual application. For example, if a virtual application uses a plug-in called custom.plugin at the version level 1.2.3.4, you cannot delete version 1.2.3.4 of custom.plugin from the system. Locking the plug-in in this way is important for the integrity and stability of virtual applications in a production environment. If the plug-in is removed and the deployed application needs to scale up or recover from a failure, the absence of the plug-in can result in application failure.

In a development environment, however, the ability to delete and replace a locked plug-in is useful because it significantly reduces the time required to test updates to a plug-in. For example, if a deployed application is using custom.plugin version 1.2.3.4 and you want to test a bug fix or new feature that you have added to the plug-in, you can import the modified custom.plugin version 1.2.3.4 plug-in and activate it on deployed virtual machines in the virtual application instead of deploying a new copy of the virtual application.

Procedure

To use the unlock for plug-in testing:

1. Update the code for the plug-in you are developing, and build the plug-in without changing the version number of the plug-in.
2. Import the plugin.unlock plug-in and then restart IBM Cloud Orchestrator.
3. Delete the existing plug-in version from the IBM Cloud Orchestrator system.
4. Import the plug-in that you updated and built in step 1.
5. Apply the changes to the virtual machine.
   a. Stop the agent and deployment inlet.
      ```
      killall java
      ```
   b. Remove installed artifacts with the following commands:
      ```
      rm -rf /opt/IBM/maestro
      cd /0config
      rm -rf 0config.log backup/ cert.tmp/ debug/ doneconfig download.zip exec_vm_tmpl/
      itlm/ lafiles/ logging/ monitor/ nodepkgs/ properties/ start/
      ```
   c. Perform any operations required to put the virtual machine in a state that is ready for a fresh activation of the plug-in. This can include stopping processes or removing files or other content that the plug-in installs on the virtual machine.
d. Restart the virtual machine and activate the plug-in with the following command:

```
/0config/0config.sh
```

The node reboots and restarts itself, as if it were a newly deployed node. It downloads the topology document, and then takes the nodeparts, parts, and roles through all the lifecycle events by running all the lifecycle startup scripts. Your newly installed version 1.2.3.4 of custom.plugin is used by the application. You can restart as many virtual machines as you need to properly test your plug-in code.

**What to do next**

If you need to update the plug-in again, you can repeat the steps to delete the installed plug-in, import the modified version, and activate the new version on the virtual machines. When you finished your testing, delete plugin.unlock and restart IBM Cloud Orchestrator.

Creating your own database plug-in:

If you want to develop a plug-in to support your own database, you can create own modeled on wasdb2 plug-in. This topic describes how the implementation of a database connection.

The wasdb2 plug-in includes parts for connections with an existing DB2, Informix, or Oracle, or connections with a pattern-deployed DB2 database. You can choose to include either or both implementations in your custom plug-in.

For a pattern-deployed database, there are two virtual machines, one running WebSphere Application Server with the user’s application, and the other running DB2 to manage data for the application. In the existing database case, the database is already up and running on another machine, either in the cloud, as a shared service, or outside the cloud managed by IBM Cloud Orchestrator. In either case, WebSphere Application Server needs to have the IP address, port number, database name, and database credentials (userid and password) to connect to the database.

In the application model, the WebSphere Application Server node and the database node are modeled as components. The link between them in the application model represents the connection between them, and provides the foundation for this data transfer of required access information.

**Pattern-deployed database connection**

Roles provide capabilities to orchestrate application startup, life-cycle management, and undeployment. For a database deployed with the IBM Web Application Pattern (not released with the product), a WAS role manages and interacts with WebSphere Application Server instance deployed on its node, and a DB2 role interacts with the DB2 instance deployed on its node. The wasdb2 plugin provides a link between the WAS and DB2 components. It inserts a dependency of the WAS role on the DB2 role. At the start of the deployment, when both the WAS and DB2 roles transition to the RUNNING state, the WAS/DB2/changed.py script runs. The DB2 role life-cycle scripts export DB2 characteristics, like hostname/IP address, port number, database name, user id and password that are required to use the DB2 database on this deployed instance. The WAS/DB2/changed.py script gets this exported data, and passes the values into wsadmin scripts to configure the information so that applications in the WebSphere Application Server node can access the database.
Roles can also contribute operations to the deployment inlet. These operations can be used to modify the running deployment. For example, you can change the password of your DB2 database. The DB2 plugin offers this operation, which changes the database password and exports this changed data. The WAS/DB2/changed.py script is notified on this update, and invokes a wsadmin script to update the changed password in WebSphere Application Server.

**Existing database connection**

Web Application Pattern (not released with the product) supports connecting to three existing database types: DB2, Informix, or Oracle. It uses a role for each: xDB2, xInformix and xOracle. These roles work like the DB2 role used for a pattern-deployed database. An application model component is available on the Pattern Builder palette for each type of database, with hostname/IP address, port number, database name, and user ID and password attributes. These values are specified at virtual application design time. A link transform makes the WAS role dependent on these roles, for existing databases. The xDB2 role start.py script exports the values that are specified in xDB2 component on the Pattern Builder pane, by using the same mechanism and key names as the DB2 role. The existing database roles offer configuration settings and deployment inlet change operations to dynamically change these configuration values just like the DB2 role. As with a pattern-deployed DB2 database, WAS/xDB/changed.py scripts get the exported xDB values (IP address, port number, database name, userid and password) and invoke appropriate WebSphere Application Server configuration scripts so that the applications on the WebSphere Application Server node can access the database.

Two attributes must be added to dependent roles.

"asDependency" : "DB"

Normally, each dependent role must provide its own dependent role scripts. The wasdb plugin provides these scripts for all databases, and they are delivered as WAS/DB scripts. While the topology document has the WAS role depending on the appropriate DB role (DB2, xDB2, xInformix or xOracle), the "asDependency" attribute maps all dependent role script calls to WAS/DB, for example for changed.py. Database dependent information, unique to each database, is passed to the wasdb link in a dblink_metadata JSONObject.

"localOnly" : true

This attribute is used in the existing resource or surrogate role cases to indicate that this role is local to the WebSphere Application Server node. It is especially important with scaling to invoke WAS/DB/changed.py only once per local WebSphere Application Server node. The next section describes surrogate roles.

The wasdb plugin contributes a WASDB link to the Web Application Pattern pattern type. The source component is a WebSphere Application Server node (vm-template). The target component is a JSONObject with two elements:

**dblink_metadata (required)**

A JSONObject with two elements:

packages (optional)

A JSONArray of package names to be installed on the WebSphere Application Server node. The packages are added in the usual way to the $sourcePackages variable in the wasdb_link.vm velocity template.

parms (required)

The database-specific parameters required by the scripts to configure Web Application Pattern to connect to the database.
role (optional)

The role to insert into the WebSphere Application Server template. It is also made a dependent role to the WAS role in the source WebSphere Application Server template, as is usual in Velocity template link transforms. The role element is used in the existing database case, for xDB2, xInformix and xOracle.

A pattern-deployed database does not need an extra role. It uses the DB2 role for the WAS role to depend on. A targetRole parameter is included in the dblink_metadata parms element for a pattern-deployed database. Its value is the name of the pattern-deployed DB role of the dependency added to the WAS role.

Using a surrogate role

The surrogate role is an option if your database pattern deploy plug-in does not provide a role that closely mimicks the DB2 role, or the data it exports does not use the same names as the DB2 role and our xDB roles. A surrogate role is added to reflect status and changes in the target database role back to the wasdb plugin in the manner it expects. For example, a database called DB3 has a DB3 role, that we want to use with our wasdb plugin. We will create a new role, DB3Surrogate, that depends on the DB3 role. It will have a DB3Surrogate/DB3/changed.py script that gets changed exported data from DB3. The WAS role will be dependent on DB3Surrogate role, which will export changed data from DB3, convert it to names and formats expected by wasdb, and export it in names wasdb expects. To realize this with WASDB link, targetRole in dblink_metadata parms would be DB3, and the DB3Surrogate role would be passed in as the role element.

Log service for plug-ins:

The logging service is a general service to collect multiple types of information. The information is securely transferred from the virtual machine and stored for review by a logging service implementation.

The information collected by the logging service is for administrative purposes and not for the application. The service can collect text and binary type file information. The file can be a single snapshot file that is never collected again or an infinitely growing file that can rotate to manage the size.

The logging service is a high-level service that supports zero to multiple registered logging service implementations. The registered implementations are the real processes that provide reports on the multiple types of information collected.

This general logging service presents a subset of the collected information in the Log Viewer page of the IBM Cloud Orchestrator user interface and the Virtual Application Console deployment Log Viewer tab.

The Log Viewer displays only the information found on the virtual machine when requested. Historical information cannot be displayed when the information is removed. For example, if the data is deleted or rolled over or if the virtual machine disappears because it is terminated. The presentation of the historical information, even after the virtual machine is no longer available, remains available to the logging service implementation to extend the Log Viewer capabilities to access the extracted information such as the external storage system.

The logging service information is explained in more detailed in the following sections:
High-level design of the log service

The logging service is automatically included on all virtual machines that have the IBM Cloud Orchestrator agent. Included is the virtual application virtual machines.

The logging service is a generic framework for plug-ins to use, notifying the logging service implementation about the multiple types of information to collect. This provides the flexibility for different logging service implementations to be registered on a single virtual machine. When notifying the logging service about the list of files to collect, specific information is required that helps all logging service implementations. The logging service uses a logtype, such as name, type, and single event pattern, and a list of specific files or directories that are pattern-based or any file created in directory, to monitor. The type and single event pattern helps the logging service understand details about the file, like how individual events are contained inside the file. The logging service provides a default list of logtypes that can be used and information how to extend the list of logtypes for the custom event patterns described as follows.

Logging service default list of logtypes

This section describes the default list of logtypes. Using these logtypes reduces the need for creating custom logtypes. These logtypes can be used by external resources to monitor the files or directories.

Table 73. Logtypes. List of logtypes and details

<table>
<thead>
<tr>
<th>Logtype name</th>
<th>Logtype details</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>&quot;description&quot;:&quot;Single file created for log entry&quot;</td>
</tr>
<tr>
<td>BinaryFile</td>
<td>&quot;description&quot;:&quot;Single binary file created for log entry&quot;</td>
</tr>
<tr>
<td>SingleLine</td>
<td>&quot;description&quot;:&quot;Each list is a new entry&quot;</td>
</tr>
<tr>
<td>MultiLineTimeStamp</td>
<td>&quot;description&quot;:&quot;Single/Multi line entry where bracket incase date/time, 10/8/10 16:42:54:109 EDT, notes the start of a new entry&quot;, &quot;start&quot;: &quot;\d{1,2}/\d{1,2}/\d{2}.<em>\d{1,2}:\d{2}:\d{2}:\d{3}.</em>\w{1,3}\&quot;*&quot;</td>
</tr>
<tr>
<td>MultiLineIP</td>
<td>&quot;description&quot;:&quot;Single/Multi line entry where IP address notes the start of a new entry&quot;, &quot;start&quot;: &quot;\d{1,3}\d{1,3}\d{1,3}\d{1,3}\d{1,3}\\\d{1,3}.*&quot;</td>
</tr>
</tbody>
</table>

Custom logtypes

External resources can create a custom logtype to assist custom event patterns in monitoring the files and directories. This is done by creating a custom JSON file packaged with the external resource. The plug-in must notify the logging service about this custom logtype file. The basic metadata fields for a logtype entry are as follows:
Table 74. Custom logtypes. List of custom logtypes and details

<table>
<thead>
<tr>
<th>Metadata field ID</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies a unique name.</td>
<td>True</td>
</tr>
<tr>
<td>description</td>
<td>Specifies a text description of the logtype</td>
<td>False</td>
</tr>
<tr>
<td>format</td>
<td>Specifies either binary or text. The default is text.</td>
<td>False</td>
</tr>
<tr>
<td>start</td>
<td>Specifies a pattern to determine where an event starts. <strong>Important:</strong> If no end tag is included, an event ends when the pattern is seen again.</td>
<td>False</td>
</tr>
<tr>
<td>end</td>
<td>Specifies a pattern to determine when an event is complete. <strong>Important:</strong> This tag determines when an event is complete, therefore, other start patterns that are found are ignored until the end pattern is found.</td>
<td>False</td>
</tr>
</tbody>
</table>

Example

"logtype-config.json" file:

```json
{"types": [  {  "name": "adaptorName2",  "description": "This is a new adaptor",  "format": "text"  "start": "\\d{2}/\w{3}/\d{4}.*\d{2}:\d{2}:\d{2}:\d{3}.*\-\d{4}\].*Start:.*",  "end": "\\d{2}/\w{3}/\d{4}.*\d{2}:\d{2}:\d{2}:\d{3}.*\-\d{4}\].*End:.*"  } ]
```

Logging service methods

Plug-in Python life cycle scripts interact with the logging service through the maestro.loggingUtil method call. The following list includes methods that the logging service utility exposes for plug-ins to call:

- maestro.loggingUtil.registerImplementation(ImplName, ImplScript)

  Allows the logging service implementations to register with the general logging service. This method indicates if the logging services are the local virtual machine implementations. When registered, the maestro.loggingUtil “monitor”, “unmonitor” and “registerPluginLogtype” calls are forwarded to the registered implementation.

  - Input: ImplName – Specifies the name of the logging service implementation.
  - Input: ImplScript – Specifies the path and file name of the Python script that is implementing the “monitor”, “unmonitor” and “registerPluginLogtype” calls.
  - Return: Specifies true if the call successfully completes or false if the call fails.

- maestro.loggingUtil.unregisterImplementation(ImplName)
Allows the logging service implementations to unregister with the general logging service. This stops the forwarding of the “monitor”, “unmonitor” and “registerPluginLogtype” calls to the service implementation.

- **Input**: ImplName – Specifies the name of the logging service implementation.
- **Return**: Specifies true if successfully complete or false if fails.

**maestro.loggingUtil.monitor(listjson)**

Allows the plug-in to notify the logging service implementations about the list of files and directories to monitor. This object indicates the specific role and lists the files to be collected. Details about the files, such as location and logtype, are provided. The location and logtype explain the structure of a single event inside the file for indexing implementations. The implementation must handle multiple duplicate invocations of this call.

- **Input**: jsonData – Specifies a JSON list that contains the details about a specific role list of information to monitor. The structure of this object is as follows: 
  
  ```json
  { "role": "<roleName>", "types": [ { "logtype": "<logtypeName>", "type": "<file or directory>", "name": "<path with or without a file name>", "pattern": "<optional with type directory noting the pattern of files inside directory>" ] } }
  ```

- **Return**: Specifies true if successfully complete or false if fails.

**maestro.loggingUtil.unmonitor(listjson)**

Allows the plug-in to remove the specified list of files and directories from being monitored. This requires the logging service implementation to make one last extraction from these files and then stop monitoring the files even if new events are logged in to the files.

- **Input**: jsonData - Specifies the JSON list that contains the details about a specific role list of information to monitor. The structure of this object is as follows: 
  
  ```json
  { "role": "<roleName>", "types": [ { "logtype": "<logtypeName>", "type": "<file or directory>", "name": "<path with or without a file name>", "pattern": "<optional with type directory noting the pattern of files inside directory>" ] } }
  ```

- **Return**: Specifies true if successfully complete or false if fails.

**maestro.loggingUtil.registerPluginLogtype(file)**

Allows the plug-in to provide information about custom logtypes if the default logtype list must be enhanced.

- **Input**: file – Specifies the JSON file that contains the definition of the log type.
- **Return**: Specifies true if successfully complete or false if fails.

**maestro.loggingUtil.isImplementationRegistered(ImplName)**

Allows plug-ins to know if a specific logging service implementation is registered on the local virtual machine.

- **Input**: ImplName – Specifies the name of the logging service implementation.
- **Return**: Specifies true or false.

**maestro.loggingUtil.getLogTypes()**

- **Input**: None
- **Return**: Specifies the JSON object of all the default logtypes that are available, represented by the default logtype-config.json file that is provided.
Plug-in interaction with the log service

A plug-in must notify the logging service with the list of directories and files to collect for the log viewer and logging service implementations.

The plug-in notifies the logging service when to start and stop monitoring the specific files. The plug-in must provide details about the files so the logging service knows the type of the file (binary or text), and the structure of a single event inside the file. The plug-in uses the logtype to describe the details of a file to the logging service so it can properly handle the events.

The plug-in uses the following methods inside the life cycle scripts any time during the life cycle execution:

- `maestro.loggingUtil.monitor(jsonData)`
- `maestro.loggingUtil.unmonitor(jsonData)`
- `maestro.loggingUtil.registerPluginLogtype(file)`
- `maestro.loggingUtil.isImplementationRegistered(ImplName)`

The following is an example of how a plug-in interacts with the logging service. The example illustrates where the default logtypes are used so that the logging service monitors both specific files and directories for historical purposes.

Inside the example plug-in, the `start.py` life cycle script first creates a JSON object that contains a list of files and directories to monitor:

```
```

Then, the example plug-in `start.py` life cycle script notifies the logging service about the list of files and directories:

```
maestro.loggingUtil.monitor(listjson)
```

The logging service monitors these specific files to display in the log viewer and allow logging service implementations configured to store these for historical purposes.

Create a log service implementation

The logging service supports custom implementations such as, log backup, logging collection and analysis service, and software monitoring like Splunk. These implementations act as the underlying process for a secure information transfer from the virtual machine and information storage for data review.

The logging service implementation is required to follow these steps:

1. Create a plug-in that contains the logging service implementation that gets registered with the logging service.
   - Because the logging service implementation can be used on virtual machines where the logging service is implemented, the implementation must be a pattern type plug-in. This allows the logging service implementation to be properly installed and managed by the plug-in infrastructure.

2. Use the plug-in life cycle script calls to register with the logging service.
The logging plug-in implementation uses its life cycle scripts to register when it is ready to receive forwarded logging service method calls. The method to do this registration is `maestro.loggingUtil.registerImplementation(ImplName, ImplScript)`.

The implementation provides a name, for example, `logbackup`, so other plug-ins required to interact with the specific implementation know if that implementation is active on the virtual machine. Also, this helps the logging service know that the services are registered. The Python script provided contains the implementation of the core forwarding methods.

When the implementation is deactivated and does not take any more forwarded calls from the logging service, the unregistered method is used. The method to do this unregistration is `maestro.loggingUtil.unregisterImplementation(ImplName)`.

This again provides the official name of the implementation.

3. Implement the core forwarding methods such as monitor, unmonitor, and registerPluginLogtype.

Each logging service implementation is required to provide a Python script that implements the following methods. These methods are automatically called when the implementation is registered. Any local plug-in on the virtual machine can call these core methods. The following are the methods to be implemented:

- **monitor(jsonData)**
  Provides the list of files and directories to be monitored with a logtype. The logtype defines the details about the binary or text file and what a single event structure looks like inside the file. If the service cares about the specific event structure, the logtype defines a generic pattern that indicates the start and end pattern of a single event for that specific file.

- **unmonitor(jsonData)**
  Provides the list of files and directories to stop monitoring.

- **registerPluginLogtype(file)**
  Provides a file that contains custom logtypes provided by a specific plug-in. This file explains unique event patterns for the specific plug-in role, for example:

```json
{"types":[
{
  "name": "DB2instance",
  "start": "------------------------------------------------------------.*",
  "end": "------------------------------------------------------------.*"
},
{
  "name": "DB2StandardLog",
  "start": \d{4}\-\d{2}\-\d{2}\-\d{1,2}\.\d{1,2}\.\d{1,6}.*"
}
]}
```

The implementation sets up these calls. For example, the `logbackup` plug-in creates a registry of files and file patterns required to regularly back up the logs to an external system.
Monitor service for plug-ins:

Plug-ins provide monitoring operations that collect and display deployment metrics for resource utilization and performance at the virtual machine, middleware, and application level.

If you are developing your own plug-ins for IBM Cloud Orchestrator, you can configure and register collectors for plug-in specific metrics at runtime and apply metadata to define the presentation of the monitoring metrics in the Virtual Application Console deployment panel.

The following illustration lists the artifacts a plug-in requires to provide monitoring.

Collector

IBM Cloud Orchestrator monitoring provides specific and built-in collectors, and generic and typed collectors. These collectors are based on an open, loosely-coupled, and collector-oriented framework. All collectors are implemented from the interface com.ibm.maestro.monitor.ICollectorService which includes the following methods:

```java
String create(JSONObject config);
JSONObject getMetadata(String uniqueId);
JSONObject getMetrics(String uniqueId);
void delete(String uniqueId);
```

IBM Cloud Orchestrator monitoring has three types of collectors that are described in the following table.

<table>
<thead>
<tr>
<th>Collector</th>
<th>Property</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.maestro.monitor.collector.itmosagent</td>
<td>Built-in</td>
<td>Collect metrics relevant to CPU, RAM, DISK and NETWORK of the virtual machine from the ITM OS Agent.</td>
</tr>
<tr>
<td>com.ibm.maestro.monitor.collector.wca</td>
<td>Built-in</td>
<td>Collect metrics relevant to CPU, RAM and STORAGE of the virtual machine from the hypervisor.</td>
</tr>
<tr>
<td>com.ibm.maestro.monitor.collector.script</td>
<td>Public</td>
<td>Collector for plug-ins that can supply metrics with shell scripts.</td>
</tr>
</tbody>
</table>

The built-in collectors are dedicated to IBM Cloud Orchestrator monitoring only. These collectors gather common metrics, like statistics at the virtual machine level. The public collectors are used by plug-in developers to interact with plug-in management facilities for metrics collecting.
Registration

To use IBM Cloud Orchestrator monitoring collectors, you must register the collectors with the plug-in configuration, providing the node, role, metrics, and collector facilities information.

IBM Cloud Orchestrator provides a Python interface to register the collectors. The definition of the interface is as follows:

```python
maestro.monitorAgent.register('{
    "node": String,
    "role": String,
    "collector": String,
    "config": JSONObject
}
)
```

The single parameter, `maestro.monitorAgent.register` is a JSON string, where:
- "node" is the name of the server running the collector
- "role" is the name of the role the collector works for
- "collector" is the collector type
- "config" is the required and optional properties of instantiating the typed collector for the specific node and role. Each type of collectors has its own "config" properties.

<table>
<thead>
<tr>
<th>Collector</th>
<th>Configuration properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.maestro.monitor.collector.itmosagent</td>
<td>{ &quot;metafile&quot;:&quot;&lt;full path to metadata json file&gt;&quot; }</td>
</tr>
<tr>
<td>com.ibm.maestro.monitor.collector.wca</td>
<td>{ &quot;metafile&quot;: &quot;&lt;full path to metadata json file&gt;&quot;, &quot;api-version&quot;: &quot;3.0&quot; }</td>
</tr>
<tr>
<td>com.ibm.maestro.monitor.collector.script</td>
<td>{ &quot;metafile&quot;: &quot;&lt;full path to metadata json file&gt;&quot;, &quot;executable&quot;: &quot;&lt;script to execute&gt;&quot;, &quot;arguments&quot;: &quot;&lt;optional arguments&gt;&quot;, &quot;validRC&quot;: &quot;&lt;optional - valid return code - defaults to 0&gt;&quot;, &quot;workdir&quot;: &quot;&lt;optional - work dir- defaults to java.io.tmpdir&gt;&quot;, &quot;timeout&quot;: &quot;&lt;optional - time out (second) - defaults to 5 seconds&gt;&quot; }</td>
</tr>
</tbody>
</table>

The following code example illustrates the registering script used in the script collector:

```python
maestro.monitorAgent.register('{
    "node": "$\{\text{maestro.node}\}$",
    "role": "$\{\text{maestro.role}\}$",
    "collector": "com.ibm.maestro.monitor.collector.script",
    "config": { "metafile": "<full path to metadata json file>",
        "executable": "<script to execute>",
        "arguments": "<arguments>",
        "validRC": "<optional - valid return code - defaults to 0>",
        "workdir": "<optional - work dir - defaults to java.io.tmpdir>",
        "timeout": "<optional - time out (second) - defaults to 5 seconds>" } }
)"
```
The registering scripts are typically put into appropriate scripts or directories of the plug-in lifecycle to ensure that the plug-in is ready to collect metrics. For example, for the WebSphere Application Server collector, the registering script is placed under the `installApp_post_handlers` directory where all scripts are executed after WebSphere Application Server is running.

**Metadata file**

The metadata file is referred to in collector registering.

The plug-in provides a JSON formatted file that includes collector metadata parameters, metric category types that it wants to expose and metadata describing each exposed metric. The content of the metadata file contains:

- Metadata file version
- array of category names to register (1..n)
- interval time in seconds to poll for updated data
- category unique configuration parameters, like `mbeanQuery`
- list of metric metadata objects
  - attributeName - Specifies an attribute from the collector to associate to this metric
  - metricName - Specifies a metric name to expose through the monitoring agent APIs
  - metricType - Specifies the data type, like range, counter, time, average, percent, and string
  - description - (optional) Specifies the string that defines the metric

The format of the metadata file is as follows:

```json
{
  "Version": <metadata file version>,
  "Category": [
    <array of category names to register (1..n)>
  ],
  "Metadata": [ ["<category name from Category[]>":{" ["updateInterval": <interval time in seconds to poll for updated data> ["metricsName": <metric name to expose through monitoring agent APIs> ["metricType": <metric value data type including "RANGE","COUNTER","TIME","AVERAGE","PERCENT","STRING"> [ , ....... ] ] } , ... ... ] }, ....... ] }
}
```

**User interface presentation**

Plug-in metrics are displayed in IBM Cloud Orchestrator Virtual Application Console Middleware Monitoring tab.

The user interface display is metadata driven. Plug-ins can provide metadata to describe the metric and category for displaying the metrics, and define the form of displaying for metrics.
monitoring_ui.json

A plug-in provides the monitoring_ui.json file for metadata. The following code is an example of the monitoring_ui.json file:

```json
[
  {
    "category": <category name from Category[] defined in metric metadata>,
    "label": <the content shown on the chart for the category>,
    "displays": [
      {
        "label": <string shown on the chart element for the metric>,
        "monitorType": <time and type properties of the metric to display>,
        "chartType": <chart type for displaying the metric>,
        "metrics": [
          {
            "attributeName": <metric name defined in the metadata>,
            "label": <string shown on the chart element for the metric>,
          }
        ]
      }
    ]
  }
]
```

### Table 77. Monitor types

<table>
<thead>
<tr>
<th>Monitor types (monitorType)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HistoricalNumber</td>
<td>Metric data in simple number for historical timeline</td>
</tr>
<tr>
<td>HistoricalPercentage</td>
<td>Metric data in percentage for historical timeline</td>
</tr>
<tr>
<td>RealtimeNumber</td>
<td>Metric data in simple number for current temporality</td>
</tr>
<tr>
<td>RealtimePercentage</td>
<td>Metric data in percentage for current temporality</td>
</tr>
</tbody>
</table>

### Table 78. Chart types

<table>
<thead>
<tr>
<th>Chart types (chartType)</th>
<th>Presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines</td>
<td></td>
</tr>
<tr>
<td>StackArea</td>
<td></td>
</tr>
<tr>
<td>StackColumn</td>
<td></td>
</tr>
<tr>
<td>Pie</td>
<td></td>
</tr>
</tbody>
</table>

The monitoring_ui.json file is located under the plugin directory of a plug-in project, for example, plugin.com.ibm.was/plugin/monitoring_ui.json. Other JSON files are also located in this directory, including config.json and config.meta.json.

### Auto scaling

The elastic scaling, or auto scaling, feature in a plug-in uses monitoring.
IBM Cloud Orchestrator auto scaling provides the run time with automatic addition or removal of virtual application and shared services instances based on instances work load.

You can optionally turn on the auto scaling feature by attaching the scaling policy to a target application or shared service. The policy is also used to deliver the scaling requirements to the backend engine. Requirements include trigger event, trigger time, and instance number, which drive the scaling procedure.

The auto scaling policy can be attached to two kinds of components in IBM Cloud Orchestrator: a virtual application and a shared service. For the virtual application, you can explicitly add the scaling policy to one or more components of the application in IBM Cloud Orchestrator’s Pattern Builder. For the shared service, the scaling policy must be described in the application model made by the plug-in developer if the service asks for the auto scaling capability.

Plug-ins, either for virtual applications or shared services, define the scaling policy, describe the policy in application model and provide transformers to explain and add scaling attributes into the topology document when the policy is deployed with plug-ins. Only if you are using shared services, can the application build automatically generate the segment of scaling policy in application model. At run time, the backend auto scaling engine first loads scaling attributes and generates the rule set for scaling trigger. Then the backend engine computes on the rule set and decides if the work load reaches a threshold for adding or removing application or share service instances. The final step of the process is to complete the request.

**Policy elements**

The auto scaling policy is composed of elements for different scaling aspects as follows:

- **Trigger Event**
  Specifies the type of monitoring metrics for the plug-in, including adding and removing plug-in instances and what thresholds they have.
  
  For each metric in the event definition, there are two thresholds: scale-in threshold and scale-out threshold. For example, the CPU utilization of virtual machines that run WebSphere Application Server instances can be the metric for the trigger event and the thresholds for scale-in and scale-out are 20% and 80%, then when the value of CPU utilization is higher than the 80%, a new WebSphere Application Server instance is launched. When the CPU utilization is below 20%, an existing WebSphere Application Server instance is selected for removal.

- **Trigger Time**
  Specifies the time it takes to hold an inspecting threshold before performing scaling operations when the threshold condition is met. For example, if trigger time is set to 120 seconds at the moment that the CPU utilization is monitored high than 80%, a timer is started. When the timer reaches 120 seconds, the scale-out operation is not started. It must be noticed that during the timing, if the CPU utilization goes out of thresholds, the timer is stopped and waits for another restarting.

- **Instance Number Range**
  Specifies the total number of instances a plug-in can have at one time or at least by scale-out or scale-in. When the cluster size of a plug-in reaches the border of its ranges, no instance is added or removed to or from the cluster, even though...
the trigger event is met. To apply the auto scaling policy to a plug-in, ensure that the scaling policy is defined in the application model that the plug-in is associated with, which collects user-specific requirement for the scaling capability. Also ensure that the policy is transformed into the topology document, which guides the backend engine to inspect trigger event and perform scaling operations.

Application model

Auto scaling capability is embodied as a policy in the application model. The application model is used to describe the components, policies, and links in the virtual applications or shared services. For virtual applications, the model can be visually displayed and edited with the Pattern Builder.

You can customize your components and policies, including the auto scaling policy, in the Pattern Builder. There is no tool to visualize shared services in the application model. Auto scaling can only be customized in the Virtual Application Console when the service is deployed. The scaling policy that is described in the application model, for either a virtual application or shared service, follows the IBM Cloud Orchestrator application model specification. The policy is defined in the node with a group of attributes.

The three auto scaling elements, trigger event, trigger time and instance number range, are described in the attribute set. There is no name convention for the attribute keys, but they must be understood by the plug-in to transfer into a topology document. The following code is an example of the elements described in the plug-in:

```json
"model": {
  "nodes": [
    {
      "id": <policy id>
      "type": <policy type>
      "attributes": {
        "No.1 metric id for trigger event": [
          < threshold for scale-in >,
          < threshold for scale-out >
        ],
        "No.1 metric for trigger event": [
          < threshold for scale-in >,
          < threshold for scale-out >
        ],
        ....... :[...... ,....... ]
    },
    "trigger time id": <trigger time value>
    "Instance range number id": [
      <min number>,
      <max number>
    ]
  }
}
```

Chapter 7. Managing and deploying virtual patterns  601
The attributes describe the scaling policy in an application model. From above JSON segment, the Trigger Event can include multiple metrics and thresholds for one scaling policy. This means that the scaling operations on a plug-in can be triggered by different condition entries with different metrics. The relationship among these entries are explicitly explained by plug-in transformer and marked in the topology document. It is not required to mark them in application model, except that their label can be used to define the relationship in the user interface. IBM Cloud Orchestrator requires the metadata be provided in a plug-in to explain components in the application model for user interface presentation. For scaling policy, the plug-in can apply proper widget types and data types to the attributes for Trigger Event, Trigger Time and Instance Number Scope.

Topology document

In the topology document, the scaling is extended to contain the attributes from auto scaling. The neat scaling contains only attributes of “min” and “max”, both of which typically have the same value. The value indicates the size of a fixed cluster on the plug-in template.

```
"vm-templates": [  
  {  
    ....  
    "scaling": {  
      "min": <number>,  
      "max": <number>  
    }  
  },  
  {  
    ....  
  }  
]
```

When additional attributes such as “triggerEvents” and “triggerTime” are included in the scaling, it evolves to an auto scaling capacity on the cluster. The value of “min” and “max” should not be the same any longer: “min” for lower limit of Instance Number Scope and “max” for the upper limit. The attributes for auto scaling are illustrated in the following JSON code example:

The attributes for auto scaling in a topology document have the name convention for keys, which are styled as bold in the previous JSON code example.

IBM Cloud Orchestrator supports multiple trigger events for a scaling operation. Those events are currently aggregated in two modes: OR and AND. The OR mode means if only one event happens, is the scaling operation triggered. The AND operation means if all events happen at the same time, only then is the scaling operation triggered. Auto scaling depends on monitoring to collect metrics for inspecting. To ensure that the right metrics are collected, the value of key "metric" in each trigger event must be consistent with the “category” and “attributeName”. These attributes are defined in the plug-in metadata for monitoring collectors. The values can be joined by “.” into “metric”. For example, “CPU.Used” represents the metric with a category of “CPU” and an attributeName of “Used”.

The transformer provided by the plug-in must define attributes of the scaling policy in the application model and map them to the named attributes in the
Topology document. The Trigger Event, Trigger Time, and Instance Number Scope auto-scaling elements correspond to "triggerEvents", "triggerTime", and "min" & "max".

**Pattern type packaging reference:**

The pattern types are a collection of plug-ins. The plug-ins contain the components, policies and links of the virtual application pattern. This topic explains the packaging of the plug-ins that create the virtual application pattern type.

Virtual application pattern types are shipped with IBM Cloud Orchestrator or they are purchasable at the IBM PureSystems Centre.

The associated plug-in files that are associated with a pattern type are as follows:

- `{ptype}.tgz` file
- `plugins/set of {plugin}.tgz` files
- `files/set of {name}.tgz` files

The `{ptype}.tgz` file is required and must contain the `patternType.json` file. The `{ptype}.tgz` file might also contain the license and localized messages. For example, the `patternType.json` file for the IBM Web Application Pattern (not released with the product) is as follows:

```json
{
    "name": "NAME",
    "shortName": "webapp",
    "version": "2.0.0.0",
    "description": "DESCRIPTION",
    "prereqs": {
        "foundation": "*"
    },
    "license": {
        "pid": "5725057",
        "type": "PVU"
    }
}
```

A pattern type defines a logical collection of plug-ins, but not the members. The members (plug-ins) define their associations with pattern types in the `config.json` file. Therefore, pattern types are dynamic collections and can be extended by third parties. For example, the `config.json` file for the DB2 plug-in (not released with the product) is as follows:

```json
{
    "name": "db2",
    "version": "2.0.0.0",
    "files": [
        "db2/db2_wse_en-9.7.0.3a-linuxx64-20110330.tgz",
        "optim/dsadm223_iwd_20110420_1600_win.zip",
        "optim/dsdev221_iwd_20110421_1200_win.zip",
        "optim/com.ibm.optim.database.administrator.pek_2.2.jar",
        "optim/com.ibm.optim.developmentstudio.pek_2.2.jar"
    ],
    "patternTypes": {
        "primary": {
            "dbaas": "1.0"
        },
        "secondary": {
            "webapp": "2.0"
        }
    }
}
```
Samples:

Use these samples to help you learn how to develop custom plug-ins. The plug-ins that you develop can be added to the IBM Cloud Orchestrator catalog and used as components, links and policies for virtual applications.

Plug-ins and pattern types are built and packaged using Apache Ant. The Plug-in Development Kit (PDK) provides Ant build (.xml) files for this purpose. These build files can run from the command line or from within Eclipse. Other development environments can work, but only command line and Eclipse are supported.

The Samples are included with the Plug-in Development Kit (PDK). Download the PDK to get started with Samples. You can also download the PDK from the IBM Cloud Orchestrator user interface Welcome page.

Attention: You must enable the PDK license before you can use the PDK. A dialogue box displays during download to assist you with the license acceptance process.

Sample pattern types and plug-ins

There are four projects in the form of plug-ins included in the PDK zip package. These plug-ins show how to design an application model, configuration, virtual machine template, and Python scripts of a plug-in.

plugin.com.ibm.sample.hellocenter: The plug-in project of HCenter plug-in

In this plug-in, you can learn how to write the lifecycle scripts, such as install, configure, start, and stop for middleware like HelloCenter. You can use the following scripts:
• install.py: Downloads artifacts from storehouse and installs the middleware. If you want to download and extend the .tgz installation file from the storage server, use the function downloadx instead.
configure.py: Downloads the artifacts uploaded by the plug-in and configures the middleware, and opens the firewall to accept customer requests. Export its IP address.

start.py: Starts the HelloCenter server and changes the role status to Running.

stop.py: Stops the HelloCenter server.

You can also access the maestro module and use the logger to log your messages. For more details about the maestro module and the advanced log such as maestro.trace_call, see the topic, Plug-in development guide.

**plugin.com.ibm.sample.hello: The plug-in project of Hello plug-in**

This plug-in accesses the HelloCenter and must work with the HClink plug-in. This plug-in contains the following scripts:

- configure.py: Logs the sender information.
- start.py: Changes the role status to Running.

**plugin.com.ibm.sample.hclink: The plug-in project of HClink plug-in**

This plug-in links Hello and HelloCenter and is installed along with the Hello plug-in in the same virtual machine. The following script is included in this plug-in:

- changed.py: This script runs only after the Hello and HelloCenter plug-in roles are in the Running state.

This script checks to see if the depended role HelloCenter exists and reads the transferred parameters from HelloCenter, which is exported in the HelloCenter configure.py script. The change.py script uses the HelloCenter IP address to access HelloCenter and prints the returned messages. This script also shows how to localize your messages.

With Hello, HelloCenter, and HClink plug-ins, you can learn the following tasks:

- How to transfer parameters between two plug-ins.
- How to make two or more plug-ins work together.

There are sample application model and configuration, virtual machine template, and virtual machine template configurations included with the plug-ins. You can design your artifacts based on the sample application model, configuration, virtual machine template and virtual machine configurations, rather than creating them from scratch. For more details, see the topic, Plug-in development guide.

**patterntypetest.hello: The pattern type project of patterntypetest.hello**

The patterntype.json file provides a sample on how to configure your pattern type.

The build.patterntype.xml file is used to build the pattern type into a package.

**Attention:** Before running this script, you must run the build.xml in the plugin.depends project to build all plug-ins in your workspace.

- The license folder includes all license documents for all supported languages.
- The locales folder stores all translated files for all supported languages for the message.json file.
Sample: Setting up the plug-in development environment:

Set up the environment for the plug-in Samples.

Before you begin

The following products are required before setting up the environment:

  - Eclipse is not required, but if you use it, use this version.
  - If you use Eclipse, you can use the Ant that comes with it. Do not install Ant separately. Ant is located in the Eclipse installation directory at eclipse/plugins/org.apache.ant_1.*.
- Java Standard Edition (SE) 6, 32-bit
- Apache Ant, 1.7 or later

About this task

This task is required before using the plug-in development Samples. To set up your environment, complete the following steps:

Procedure

Import the sample source code and build the hello pattern type binary package.

1. Create a workspace called iwd-pdk-workspace in Eclipse.
2. Download the pdk.sample_<version>.zip to a file directory.
3. Click File > Import > General > Existing Projects into Workspace.
4. Click Select archive file and click Browse…. Select the iwd-pdk-workspace directory where you downloaded the pdk-<version>.zip file.
5. Select the plugin.depends project and the four sample projects.

After the import is done, the following projects are located in your workspace:

- patterntypetest.hello
- plugin.com.ibm.sample.hclink
- plugin.com.ibm.sample.hello
- plugin.com.ibm.sample.hellocenter
- plugin.depends

These projects are used in the next steps of end to end process of using custom developed plug-ins.

Results

You have set up your plug-in development environment.

What to do next

Develop a plug-in and pattern type.
Sample: Developing a plug-in and pattern type with the Eclipse Framework:

This topic is an example of how to create a plug-in and pattern type using the Eclipse Framework.

**Before you begin**

Download and install the Plug-in Development Kit (PDK) and set up the development environment.

**About this task**

**Procedure**

1. Go to the workspace that you created in the topic, *Setting up the samples environment section*.
2. Build a single plug-in.
3. In the plugin.depends project, run the build.xml Ant script. To run the Ant script, right-click on the file and select Run As > Ant Build. The plug-in build process starts.
   
   When the build process completes, refresh the project and a folder named export displays. All of the build artifacts are listed in the export folder.

   The plug-in package is in the root of export folder.
4. Build all plug-ins in the workspace. Select the build.xml file in the root of the plugin.depends project. Right-click and select Run As > Ant Build. The plug-in build process starts.
   
   When the build process completes, refresh the project. A folder named image displays in the sub-folder plug-ins, where all of the built plug-in packages are located.
5. Build a single pattern type. Before this step, you must successfully complete Step 2.
   
   Select the build.patterntype.xml file in the root of the patterntypetest.hello project. Right-click and select Run As > Ant Build. The pattern type build process starts.

   When the build process completes, refresh the project. A folder named export displays where all the build artifacts are listed. The pattern type package is in the root of export folder.
6. Navigate to the root of the export folder. The hello-2.0.0.0.tgz pattern type binary file is located here. The sample pattern type patterntypetest.hello release package includes three plug-ins:
   
   - HCenter plug-in: This plug-in operates a simple message center middleware named HelloCenter. HelloCenter opens port 4000 and listens to client requests, and generates and returns greeting messages.
   
   - Hello plug-in: This plug-in is a client component of HelloCenter. It sends a request with the message sender identity to the Hello Center and tries to get the returned greeting message and display the message on the console.
   
   - HLink: This link plug-in from Hello to HCenter specifies the receiver name of greeting message.

**Results**

You have created a plug-in that can be imported into the IBM Cloud Orchestrator catalog.
What to do next

Import the plug-in into IBM Cloud Orchestrator.

Sample: Import a plug-in and pattern type into IBM Cloud Orchestrator:

This topic is an example of how to import your sample plug-in into IBM Cloud Orchestrator.

Before you begin

Develop the plug-in.

About this task

Procedure

Import the pattern type hello-2.0.0.0.tgz and try sample plug-ins. The steps for this task are based on a scenario where you have installed and configured IBM Cloud Orchestrator.

1. Import pattern type patternpetest.hello.
   a. Log in into IBM Cloud Orchestrator as administrator or as a user with permission to create a new pattern type.
   b. Click PATTERNS > Deployer Configuration > Pattern Types.
   c. Click the New icon (+).
      The Install a pattern type window displays.
   d. On the Local tab, click Browse.
      Select the hello-2.0.0.0.tgz file. When the installation process completes, the pattern type, patternpetest.hello, is displayed in the Pattern Types palette.
   e. Select patternpetest.hello pattern type from the drop down list
      The pattern type details display on the right.
   f. Click Accept to accept the license. Click Enable to enable this pattern type.

Results

You have imported a plug-in and pattern type into IBM Cloud Orchestrator catalog.

What to do next

Create an application that can be deployed to IBM Cloud Orchestrator.

Sample: Creating an application with the patternpetest.hello plug-in:

In this Samples topic, you use the patternpetest.hello plug-in to create an application that is deployed to IBM Cloud Orchestrator.

Before you begin

Set up your development environment and develop a plug-in and pattern type.
About this task

To create an application with the plug-ins, complete the following steps:

Procedure

1. Click **PATTERNS > Pattern Design > Virtual Application Patterns**. The Virtual Application Patterns palette displays. Ensure that the pattern=pattern_test_hello 2.0 is selected.
2. Select the **pattern=pattern_test_hello 2.0** from the patterns list.
3. Click the **New** icon (+) at the top of the page. The Create Application dialogue window displays.
4. Select the type of application to create.
5. Click **Start Building**. The Pattern Builder displays in a new window.
6. Select the **Diagram** tab.
7. In the **Other Components** located in the left palette, drag the Hello and HelloCenter components into the middle section of the canvas. Name the application to the right in the Virtual Application palette.
8. Select the **List View** tab (located next to the Diagram tab).
9. Create the **sample_userlist.json** file with the following content:

   ```json
   ["Mike","Alice","Joe"]
   ```

10. Click the HelloCenter component. In the right attributes view, type **sample** in the **Hello Center Name** attribute field. Upload the created **sample_userlist.json** file in the **Registered User List**.
11. In the **Diagram** tab, click the Hello component. In the right attributes view, type one of the following: **"Mike","Alice","Joe"**.
12. Link Hello to HelloCenter. Click the link and type any name in the receiver of greeting message attribute field.
13. To review all of the settings configured in the previous steps, click the **List View** tab on the left palette.
14. Click **Save** and return to the Virtual Application Patterns page.
15. Refresh the Virtual Application Patterns page to display the new application. The application displays in the list located on the left.
16. Select the application and click the **Deploy** icon in the top right palette.

   The Deploy Virtual Application window displays. Complete the settings information and click **OK**. For more information about deploying virtual applications, see the topic, [Deploying virtual applications](#). When the deployment is complete, the status icon turns green.
17. To view the deployed application, click **PATTERNS > Instances > Virtual Application Instances**. The Virtual Application Instances window displays.
18. Select the **virtual_application_instance**. The details view displays to the right.
19. Find the row of **Hello_Plugin-HVM-XXXX**, where **XXXX** is a set of digits in the virtual machine list.
20. Click the log to the right of **Running**. The Log page displays.
21. In this log page, from the root, go to **IWD Agent > "./logs/Hello_Plugin-HVM. XXXX.hello" > console.log**.

The following information is provided:
Results

You have created a virtual application and deployed it to IBM Cloud Orchestrator.

What to do next

Monitor the virtual application instance.

Sample: Creating a plug-in project from the command line:

This topic is an example of how to create a plug-in project using command-line tools.

Before you begin

1. Open the command-line tool.

2. cd to the plugin.depends project directory in your workspace.

3. Set the ANT_HOME environment variable. You can use Ant in your Eclipse installation at eclipse/plugins/org.apache.ant_1.7*. You can also invoke this Ant script from Eclipse. To do this, right-click create.plugin.project.xml in the plugin.depends project. Select Run As > Ant Build. Click the Main tab. In the argument section, type the various -Dproject.name=jp1 values provided in this sample.

About this task

Continue with this task by completing the following steps:

Procedure

1. Create a template plug-in project as follows:
   ant -Dproject.name=tpl -Dplugin.name=a.b.c.template -f create.plugin.project.xml

   project.name property is optional and if it is not specified, it will default to the value of the plugin.name.

2. Create a Java plug-in project as follows:
   a. Create a Java plug-in project that contains no package name:
      (Java assumed on java classname)
      ant -Dproject.name=jp1 -Dplugin.name=a.b.c.java -Djava.classname=MyPlugin
      -f create.plugin.project.xml
   b. Create a Java plug-in project that contains a package name:
      ant -Dproject.name=jp2 -Dplugin.name=a.b.c.java -Djava.classname=a.b.c.MyPlugin
      -f create.plugin.project.xml

3. Verify that the command is successful. Import the newly created projects into your workspace.

To build the plug-in projects, for example, jp1, you can find build.plugin.xml in project jp1. Right-click build.plugin.xml and issue Run As > Ant Build
with the goal clean, publish selected. The equivalent Ant command is to issue
the following command in the project jp1 directory:

.ant -f build.plugin.xml clean publish

The plug-in a.b.c.java-<version>.tgz is created in the export directory.

Sample: Building a single plug-in and pattern type with the command-line tools:

This topic is an example of how to create a plug-in and pattern type using
command-line tools.

Before you begin

This task assumes that you have the following installed:
- The Ant build environment version 1.7.1 or higher.
- A command-line environment like Linux console or Windows command line
  interface.
- A message format tool such as msgfmt (Linux) or msgfmt.exe (Windows). Add
  the tool folder into the system path to ensure that you can start the tool without
  the full path.

Procedure
1. Go to the workspace that you created in the topic, Setting up the samples
   environment section.
2. Navigate to the root folder of target plug-in project.
3. Type this command to build a single plug-in:
   <ant command path> -f build.plugin.xml

   The building information displays in the console.
4. Go to the export folder of the plug-in project. This folder is generated by step
   3. Locate the plug-in package, which is a .tgz file.
5. Navigate to the root of the plugin.depends project.
6. Type the following command to build all plug-ins in this workspace:
   <ant command path> -f build.xml

   This command builds the plug-ins in this workspace one at a time. After the
   script starts, go to the image/plugins folder of the plugin.depends project to
   check all of the built plug-in packages.
7. Navigate to the root of the pattern type project, patterntypetest.hello, and type
   the following command:
   <ant command path> -f build.patterntype.xml

   After the script starts, go to the root of the export folder of the
   patterntypetest.hello project to check the built pattern type package, which is a
   .tgz file.
Deploying virtual applications

A *virtual application* is an application that has been optimized to run on a virtual environment. The application software and middleware is located inside a virtual machine in a manner that maximizes the performance of the application. By keeping the system software to a minimal set of packages required to support the application, the maintenance and administration of the virtual application is reduced. The *virtual application pattern* is the first step to building a virtual application.

About this task

The *virtual application template* and a *virtual application pattern* are the first steps to building a virtual application.

You can use virtual application template that is associated with a virtual application pattern to start building an application.

The virtual application pattern is a collection of plug-ins. The plug-ins in a virtual application pattern define a complete set of platform resources to fulfill a business need, including web applications, databases, user registries, and more. These components or plug-ins can be added to a virtual application pattern using the Pattern Builder or they might already exist in the pattern.

A deployed virtual application is called a *virtual application instance*.

Procedure

1. Deploy a virtual application using one of the following methods:
   - [From a virtual application pattern](#)
   - [From a virtual application template](#)
2. Secure a virtual application
3. Monitor and administer virtual applications

Deploying virtual application patterns

After you create a virtual application, you can provision and deploy it on the infrastructure cloud. Each deployment of a virtual application represents a running *virtual application instance* on the cloud infrastructure. You can deploy a given virtual application multiple times. When you deploy your virtual application you can also configure Secure Shell (SSH) key-based access.

Before you begin

Set the deploy base image by performing the following steps:

1. Add the base image into the catalog. For information about adding images, see [Importing a virtual image to the catalog](#) and [Registering a virtual image](#).
2. Set the base image as the default. Click PATTERNS > Deployer Configuration > Default Deploy Settings». The Settings for Default Deploy window is displayed.
3. Click «Change» and select an image.
4. Click Update to save your changes.

Create a virtual application.
About this task

When a virtual application is deployed, the Pattern Builder allocates necessary resources, such as virtual machines and block storage on the cloud infrastructure, and deploys, configures, and starts the virtual application components in the cloud.

Policies that are associated with the virtual application typically influence how cloud infrastructure resources and virtual application pattern components are allocated for a given deployment. For example, a single virtual machine running the web application is provisioned when a web application component is deployed. However, a scaling policy that is associated with a web application results in multiple virtual machines, equal to the cluster size that you specify for the scaling policy, provisioned for the web application, an elastic load balancer cloud component that is used for routing HTTP requests, and a set of WebSphere Extreme Scale application components that facilitate session replication across the cluster members of the web application.

The time it takes to deploy a virtual application depends on several factors, such as the size of the virtual application pattern parts and the inter-dependencies of parts in the application definition, network usage, storage usage, and the provisioning speed of the virtual machine on the cloud infrastructure.

You can add SSH key-based access to your workload virtual machine when deploying the virtual application. This type of security provides better protection than password-based access.

To deploy a virtual application pattern:

Procedure

1. Click PATTERNS > Pattern Design > Virtual Application Patterns. The Virtual Application Patterns palette displays.
2. From the left panel of the Virtual Application Patterns palette, select the virtual_application that you want to deploy.
3. Click the Deploy icon to deploy the virtual application. The Deploy Virtual Application dialogue box displays.
4. Complete the Target environment profile fields.
   These settings provide deployment configuration information, like virtual machine names, IP address assignment, and cloud groups. Deploying virtual applications with environment profiles enables deployments across tiers from a single application.
   a. Select the IP type filter, IPv4 or IPv6, in the Filter by IP type field.
   b. Select the Filter by profile type from the drop down menu.
   c. Select the Profile from the drop down menu.
   d. Select the Cloud group from the drop down menu.
   e. Select the IP group from the drop down menu.
5. Click Advanced to set the SSH public key.
   a. Enter an SSH public key in the SSH Key field. You can use a text editor to open your public key file and, copy and paste the key into the SSH Key field.
**Important:** Do not use `cat / less` to copy and paste from the user interface. This type of cut and paste introduces spaces to the key and you cannot gain access to the virtual machine.

If you do not want to use an existing SSH public key, you can generate one as described in the next step.

a. Click **Generate** to generate the key. The SSH key is automatically generated in the SSH Key field. Select **Click here to download the file containing the private key** to save the private key file. Save the file to a secure location and you can name the key. The default name is `id_rsa.txt`. The system does not keep a copy of the private key. If you do not download the private key, you cannot gain access to the virtual machine, unless you generate a new key pair again using user interface. You can also copy and paste the public key, save the key, and reuse the same key pair for another deployment. When you have the private key, make sure that it has the correct permissions (`chmod 0400 id_rsa.txt`). By default, the ssh client does not use a private key file with wide open permission.

6. Click **OK**. A message displays at the top of the Pattern Builder confirming that the virtual application is in the deployment process. You can also check the status of the deployment from this message.

**Attention:** You cannot modify a virtual application after you deploy it. You must stop the deployed virtual application before you can change it.

When the virtual application is deployed, to view the virtual instance, click **PATTERNS > Instances > Virtual Application Instances**. The Virtual Application Instances palette displays.

**Note:** If the deployment process is stopped and in the Virtual Application Instances window the `flavor` error message is displayed, you must create a flavor with memory, vCPUs, and storage values equal to or greater than the requirements of the virtual machines that you are deploying. For information about creating flavors, see “Creating new flavors in OpenStack” on page 93.

After you created the new flavor, delete the virtual application instance and redeploy the virtual application pattern.

7. View the details of the deployed virtual application in the Virtual Application Instances palette. The details include a list of virtual machines provisioned on the cloud infrastructure for that deployment, the IP address, virtual machine status, and role status. **Role** is a unit of function that is performed by the virtual application middleware on a virtual machine.

The status values are listed in the following table:

<table>
<thead>
<tr>
<th>Status</th>
<th>Deployment description</th>
<th>Virtual machine description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAUNCHING</td>
<td>The virtual application is being deployed.</td>
<td>The virtual machine is being provisioned on the infrastructure cloud.</td>
</tr>
<tr>
<td>INSTALLING</td>
<td>Not applicable</td>
<td>The components of the virtual application are being provisioned on the virtual machine.</td>
</tr>
</tbody>
</table>
Table 79. Possible status values for a deployed virtual application (continued)

<table>
<thead>
<tr>
<th>Status</th>
<th>Deployment description</th>
<th>Virtual machine description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNNING</td>
<td>Resources are being provisioned on the infrastructure cloud.</td>
<td>The components of the virtual application are running on the virtual machine and can be accessed.</td>
</tr>
<tr>
<td>TERMINATED</td>
<td>The deployment process is stopped.</td>
<td>The virtual machine is stopped.</td>
</tr>
<tr>
<td>FAILED</td>
<td>The deployment process could not be started due to either the application configuration or a failure occurring in the infrastructure cloud.</td>
<td>The virtual machine did not start successfully.</td>
</tr>
</tbody>
</table>

You can also view the virtual machine role health status information. For example, a red check mark is located on the green status arrow when the CPU is critical on the virtual machine.

Click Endpoint to view the endpoint information for a given role. For a deployment with DB2 you can have more than one endpoint, for example, an endpoint for application developer one for database administrator.

Results

Your virtual application instance is successfully deployed and started. To stop the virtual application instance, select the application from the list, and click Stop.

To redeploy a virtual application, select the virtual application from the Virtual Application Patterns palette, and click the Deploy icon in the Pattern Builder.

To remove a stopped application, select it from the Virtual Application Patterns palette, and click Delete.

What to do next

After you deploy your virtual application, you can use the IP address of the virtual machines to access the application artifacts. For example:

To gain access to your virtual machine after deployment type:

```
ssh -i id_rsa.txt virtuser@<your_workload_ip>
```

For SCP:

```
scp -i id_rsa.txt myfiles.txt virtuser@<your_workload_ip>
```

You can now log in to your virtual machine without a password.

To gain root access:

```
sudo su -
```

To run a command with root access:

```
sudo /sbin/ifconfig
```
You can view and monitor statistics for your deployed virtual machines and download and view the log files from the user interface. For more information, see “Monitoring virtual application instances” on page 626.

Deploying virtual application templates
You can directly deploy a virtual application template from the IBM Cloud Orchestrator catalog. Virtual application templates that are associated with pattern types are shipped with the product or you can create one.

Before you begin
You must have a virtual application template created or use a preinstalled virtual application template.

About this task
You can use the Pattern Builder to allocates necessary components, links and policies to application template.

To deploy a virtual application template:

Procedure
1. Click PATTERNS > Pattern Design > Virtual Application Templates. The Virtual Application Templates palette displays.
2. From the left panel of the Virtual Application Templates palette, select the virtual_application_template that you want to deploy.
3. Click the Deploy icon. The Deploy Virtual Application dialogue box displays.
4. Complete the Target environment profile fields.
   These settings provide deployment configuration information, like virtual machine names, IP address assignment, and cloud groups. Deploying virtual applications with environment profiles enables deployments across tiers from a single application.
   a. Select the IP type filter, IPv4 or IPv6, in the Filter by IP type field.
   b. Select the Filter by profile type from the drop down menu.
   c. Select the Profile from the drop down menu.
   d. Select the Cloud group from the drop down menu.
   e. Select the IP group from the drop down menu.
5. Click Advanced to set the SSH public key.
   a. Enter an SSH public key in the SSH Key field. You can use a text editor to open your public key file and, copy and paste the key into the SSH Key field.

   Important: Do not use cat / less to copy and paste from the user interface. This type of cut and paste introduces spaces to the key and you cannot gain access to the virtual machine.
   If you do not want to use an existing SSH public key, you can generate one as described in the next step.
   a. Click Generate to generate the key. The SSH key is automatically generated in the SSH Key field. Select Click here to download the file containing the private key->Save to save the private key file. Save the file to a secure location and you can name the key. The default name is id_rsa.txt. The system does not keep a copy of the private key. If you do not download the private key, you cannot gain access to the virtual machine, unless you
generate a new key pair again using user interface. You can also copy and paste the public key, save the key, and reuse the same key pair for another deployment. When you have the private key, make sure that it has the correct permissions (chmod 0400 id_rsa.txt). By default, the ssh client does not use a private key file with wide open permission.

6. Click OK. A message displays at the top of the Pattern Builder confirming that the virtual application is in the deployment process. You can also check the status of the deployment from this message.

**Attention:** You cannot modify a virtual application after you deploy it. You must stop the deployed virtual application before you can change it.

When the virtual application is deployed, to view the virtual instance, click **PATTERNS > Instances > Virtual Application Instances.** The Virtual Application Instances palette displays.

7. View the details of the deployed virtual application in the Virtual Application Instances palette. The details include a list of virtual machines provisioned on the cloud infrastructure for that deployment, the IP address, virtual machine status, and role status. Role is a unit of function that is performed by the virtual application middleware on a virtual machine.

The status values are listed in the following table:

<table>
<thead>
<tr>
<th>Status</th>
<th>Deployment description</th>
<th>Virtual machine description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAUNCHING</td>
<td>The virtual application is being deployed.</td>
<td>The virtual machine is being provisioned on the infrastructure cloud.</td>
</tr>
<tr>
<td>INSTALLING</td>
<td>Not applicable</td>
<td>The components of the virtual application are being provisioned on the virtual machine.</td>
</tr>
<tr>
<td>RUNNING</td>
<td>Resources are being provisioned on the infrastructure cloud.</td>
<td>The components of the virtual application are running on the virtual machine and can be accessed.</td>
</tr>
<tr>
<td>TERMINATED</td>
<td>The deployment process is stopped.</td>
<td>The virtual machine is stopped.</td>
</tr>
<tr>
<td>FAILED</td>
<td>The deployment process could not be started due to either the application configuration or a failure occurring in the infrastructure cloud.</td>
<td>The virtual machine did not start successfully.</td>
</tr>
</tbody>
</table>

You can also view the virtual machine role health status information. For example, a red check mark is located on the green status arrow when the CPU is critical on the virtual machine.

Click **Endpoint** to view the endpoint information for a given role. For a deployment with DB2 you can have more than one endpoint, for example, an endpoint for application developer one for database administrator.
Results

Your virtual application instance is successfully deployed and started. To stop the virtual application instance, select the application from the list, and click Stop.

To redeploy a virtual application, select the virtual application from the Virtual Application Patterns palette, and click the Deploy icon in the Pattern Builder.

To remove a stopped application, select it from the Virtual Application Patterns palette, and click Delete.

What to do next

After you deploy your virtual application, you can use the IP address of the virtual machines to access the application artifacts. For example:

To gain access to your virtual machine after deployment type:

```
ssh -i id_rsa.txt virtuser@<your_workload_ip>
```

For SCP:

```
scp -i id_rsa.txt myfiles.txt virtuser@<your_workload_ip>
```

You can now log in to your virtual machine without a password.

To gain root access:

```
sudo su -
```

To run a command with root access:

```
sudo /sbin/ifconfig
```

You can view and monitor statistics for your deployed virtual machines and download and view the log files from the user interface. For more information, see “Monitoring virtual application instances” on page 626.

Securing virtual applications

This topic introduces the security levels available to virtual applications. You can secure the virtual application at the component level or as a virtual application instance.

About this task

The following are levels of security used in virtual applications:

Procedure

-  "User roles in IBM Cloud Orchestrator" on page 206. Review the user roles that apply to creating a virtual application pattern.
-  Secure Socket Layer (SSL)
-  Configuring Secure Shell (SSH) key-based access during application deployment
-  "Configuring SSH key-based access in the user interface" on page 622
-  LTPA keys
Securing virtual application instances with SSL:

The security of a virtual application instance can be managed with Secure Sockets Layer (SSL) certificates.

About this task

You can manage personal and application SSL certificates in WebSphere Application Server keystore and truststore for secure inbound authentication and communication. To ensure SSL communication, servers require a personal certificate that is either self-signed, chained or signed by an external certificate authority (CA).

IBM Cloud Orchestrator supports the use of one certificate per application deployment. By default, your deployed application has a uniquely generated certificate signed by the internal WebSphere root signer certificate. The certificate is valid for one year, but can be renewed at any time. The IBM Cloud Orchestrator also supports replacing this default certificate with a certificate signed by an external CA.

The SSL functionality is a simplified security layer on top of, and subset of, what is provided by WebSphere Application Server.

Procedure

1. Click PATTERNS > Instances > Virtual Application Instances. The Virtual Application Instances palette displays.
2. Select the WebSphere Application Server virtual application instance. The virtual application instance details display.
3. Click the Manage icon.
4. Select the Operations tab. The Operations palette opens.
5. Select the WebSphere Application Server application. The deployment operations palette displays the operations on the right. Now you can manage the SSL certificates.
6. Renew the WebSphere Application Server application SSL certificate. To renew the default application certificate, expand Renew WebSphere Application Server application SSL certificate to display the deployment operations palette. Click Submit. The operation status displays in the Operation Execution Results palette. When the operation displays as successful the validity of the certificate is extended. You might need to wait a short time for the updated certificate to propagate to all of the application servers if your deployment has multiple WebSphere Application Server nodes.

   Important: This operation is only possible if you have not previously replaced the default application certificate with a CA-signed certificate; it otherwise fails.
7. Configure a CA-signed certificate.
   a. Create a signer request. To replace the default application certificate with one signed by an external CA, you must first create a personal certificate request. To create a persona certificate request, expand Create CA signer in the deployment operations palette and complete the fields. The common name (CN) matches the domain name that is used for the application. Click Submit.
When the operation is complete, the status shows as successful in the Operation Execution Results palette. A link is available in the **Return Value** column to download the newly generated signer request file. You must provide this file to the CA for signing.

**Important:** IBM Cloud Orchestrator supports only one pending certificate signer request. If a new request is created before accepting the CA-signed certificate corresponding to a previous request, that certificate is no longer accepted.

b. **Upload a CA-signer request.**

This step is used to import a CA-signed certificate. The signed certificate must correspond to the last signer request created for the application deployment in question. Once imported, the certificate replaces the previously configured application certificate, whether that is the default WebSphere signed certificate or a previously imported CA-signed certificate. To import, click **Browse** to locate the certificate file. The certificate file must be in base64-encoded PEM form. A progress bar shows that the file is uploading. When the upload completes, click **Submit** to complete the operation. Success is indicated in the Operation Execution Results palette.

8. **Manage the truststore certificate.**

a. **Import a WebSphere Application Server truststore certificate.** Use these instructions to import external signer certificates. The certificate file must be in base64-encoded PEM form. When uploading a certificate to import, a unique alias must be specified to identify the certificate. Make sure to record the chosen alias since there is not currently a way to list previously imported certificates. There is no limitation on the number of certificates that can be imported and reside in the truststore concurrently.

For applications that are deployed on IBM Cloud Orchestrator to securely connect to external services over SSL the appropriate signer (public) certificates must reside in the WebSphere Application Server truststore.

b. **Remove a WebSphere Application Server truststore certificate.** You can remove previously imported signer certificates from the truststore. You must provide the alias given when the certificate was imported.

9. **Export a certificate.** You can also extract and export the signer part of the current application SSL certificate or the WebSphere root signer certificate from the deployment operations screen. Expand **Export certificates** and select a certificate that you want to export from the drop-down menu (**Application certificate** or **WebSphere Application Server root signer certificate**), and click **Submit**. When the certificate is ready, the operation status displays as successful in the Operation Execution Results palette. You can download the file using the link in the **Return Value** column. The certificate file is a base64-encoded PEM form.

**Results**

You have performed various tasks to manage the SSL truststore and keystore certificates.
Configuring SSH key-based access during application deployment:

You can add Secure Shell (SSH) key-based access to your workload virtual machine when deploying the virtual application pattern. This type of security provides better protection than password-based access.

Before you begin

You must create the virtual application template and pattern before you can apply SSH.

About this task

When deploying a virtual application pattern, you can configure an RSA key pair to enable SSH key based access to the virtual machines. You can either provide your own externally-generated public key or allow the system to generate a new key pair for you.

Procedure

1. Select the **Advanced** check box to add the SSH protocol key in the Deploy Virtual Application dialog box.
2. Enter your SSH public key in the **SSH Key** field. You can use a text editor to open your public key file and, copy and paste the key into the SSH Key field. The key string must be in the public key format used in the OpenSSH authorized_keys file.

   **Important:** Do not copy the key from the console output of the Linux command `more`. This can introduce line breaks into the key that might render it invalid.

   If you do not want to use an existing SSH public key, you can generate one as described in the next step.

   a. Click **Generate** to generate the key. The SSH public key is automatically populated in the SSH Key field. Select **Click here to download the file containing the corresponding private key** -> **Save** to save the private key file. Save the file to a secure location and you can name the key. The default name is `id_rsa`.

   **Attention:** If you are using Windows, the downloaded file might be renamed to `id_rsa.txt`. The system does not keep a copy of the private key. If you do not download the private key, you cannot gain access to the virtual machine, unless you configure a new public key through the deployment management user interface.

   The generated key pair can be reused for subsequent deployments, which means that your SSH client does not have to be reconfigured with a new private each time. Copy and save the generated public key from the SSH Key field and paste it into the same field the next time you deploy a virtual application.

   The generated `id_rsa` private key file is in OpenSSH format and can be used with OpenSSH (the standard Linux SSH implementation) and is compatible with SSH clients. To use the key with other clients, the key might need to be converted to a different format. If this cannot be done, it is recommended that you generate a key pair separately.
**Important:** It is highly recommended that you save `id_rsa` in a secure location.

b. Click **OK** to continue to deploy your virtual application with SSH enabled. A message displays at the top of the Pattern Builder confirming that the virtual application is in the deployment process. You can also check the status of the deployment from this message.

c. To log in to the operating system of your virtual application virtual machines, find the appropriate IP address from the Virtual Application Instance details palette and specify the user ID, “virtuser”, along with the private key. When you are connected, the user ID can be used for root-level access without requiring a password.

**Remember:** To use the private key with the OpenSSH client (Linux SSH command), make sure that it has the correct permissions (chmod 0400 `id_rsa`). By default the SSH client rejects a private key file with less restrictive permissions.

**Results**

The virtual application pattern for which you are deploying has SSH key-based access enabled.

**Attention:** You can also generate an SSH key when you configure shared services. See the topic, Working with shared services, for more information.

**Configuring SSH key-based access in the user interface:**

You can update a Secure Shell (SSH) key-based access configuration for a deployed virtual instance by accessing the **Operations** tab in the Virtual Application Console of IBM Cloud Orchestrator.

**Before you begin**

You must have **admin** role to complete this task.

You must also create the virtual application template and pattern before you can apply SSH.

**About this task**

When deploying a virtual application pattern, you can configure an RSA key pair to enable SSH key based access to the virtual machines. You can either provide your own externally-generated public key or allow the system to generate a new key pair for you.

**Procedure**

1. Click **PATTERNS > Instances > Virtual Application Instances**. The Virtual Application Instances palette displays with the virtual application instances listed.
2. Select a `virtual_application_instance`. The details display to the right.
3. Click the **Manage** icon in the upper right corner.
4. Click **Operations**. The Operations palette displays.
5. Click **SSH**. You can add or update a public key and remove VM SSH public keys from this palette.
6. To upload a SSH public key, copy and paste your SSH key in the Public Key field and click Submit in the Add or update VM SSH public key section. If you already have a public key in the Public Key field, the key is replaced.

   Important: Do not copy the key from the console output of the Linux command more. This can introduce line breaks into the key that might render it invalid.

7. To remove VM SSH public keys, click Submit in the Remove VM SSH public keys section.

Results

You have added a new VM SSH public key or removed VM SSH public keys.

LTPA keys:

Lightweight Third-Party Authentication (LTPA), is an authentication technology used in the web application that is deployed into the cloud infrastructure.

About this task

There are various tasks you can do with LTPA keys, including regenerating keys, importing keys and exporting keys.

To configure the LTPA keys, follow these steps:

Procedure

1. Click PATTERNS > Instances > Virtual Application Instances. The Virtual Application Instances palette displays.
2. Select the WebSphere Application Server virtual application instance. The virtual application instance details display.
3. Click the Manage icon.
5. Select the WebSphere Application Server application. The deployment operations palette displays the operations on the right. Now you can manage the LTPA keys.
6. Regenerate LTPA keys. To regenerate LTPA keys, expand Regenerate LTPA keys in the Deployment operations palette. Click Submit. A confirmation dialogue window displays. Click Yes to confirm that you want to regenerate the LTPA keys.

   The operation status displays in the Operation Execution Results palette. When the operation displays as successful, the LTPA keys are regenerated.
7. Import LTPA keys. To import an LTPA key, expand Import LTPA keys in the Deployment operations palette. Click Browse to locate the LTPA key that you want to import. Click Submit. A confirmation dialogue window displays. Click Yes to confirm that you want to import the LTPA key. The operation status displays in the Operation Execution Results palette. When the operation displays as successful the LTPA key is imported.
8. Export LTPA keys. To export LTPA keys, expand Export LTPA keys in the Deployment operations palette. Click Submit. The operation status displays in the Operation Execution Results palette. When the operation displays as successful the LTPA key is exported. The exported key can be downloaded through the link that is listed in the operation status section.
Working with virtual application instances

Each deployment of a virtual application represents a running virtual application instance on the cloud environment. You can view and monitor deployed virtual application instances from the Virtual Application Console.

Before you begin

Deploy a virtual application from IBM Cloud Orchestrator to the cloud environment.

About this task

To manage a virtual application instance:

Procedure

1. Click PATTERNS > Instances > Virtual Application Instances. The Virtual Application Instances palette displays. The virtual application instances are listed by name. You can sort the list by application name or sort by status.
   
   Attention: You can also view the virtual application instances on a page where all instances, such as virtual appliance instances, virtual system instances, shared services instances and database instances are listed. Click PATTERNS > Instances > Virtual Application Instances. Every instance running on the IP address are listed.

2. From the left panel of the Virtual Application Instances palette, select the virtual_application_instance that you want to review.

3. The Maintain view displays to the right. The details include a list of virtual machines provisioned on the cloud infrastructure for that deployment, the IP address, virtual machine status, and role status. Role is a unit of function that is performed by the virtual application middleware on a virtual machine.

   The status values are listed in the following table:

<table>
<thead>
<tr>
<th>Status</th>
<th>Deployment description</th>
<th>Virtual machine description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAUNCHING</td>
<td>The virtual application is being deployed.</td>
<td>The virtual machine is being provisioned on the infrastructure cloud.</td>
</tr>
<tr>
<td>INSTALLING</td>
<td>Not applicable</td>
<td>The components of the virtual application are being provisioned on the virtual machine.</td>
</tr>
<tr>
<td>RUNNING</td>
<td>Resources are being provisioned on the infrastructure cloud.</td>
<td>The components of the virtual application are running on the virtual machine and can be accessed.</td>
</tr>
<tr>
<td>TERMINATED</td>
<td>The deployment process is stopped.</td>
<td>The virtual machine is stopped.</td>
</tr>
<tr>
<td>FAILED</td>
<td>The deployment process could not be started due to either the application configuration or a failure occurring in the infrastructure cloud.</td>
<td>The virtual machine did not start successfully.</td>
</tr>
</tbody>
</table>
4. View the virtual machine role health status information. For example, a red
check mark is located on the green status arrow when the CPU is critical on the
virtual machine.

5. Click **Endpoint** to view the endpoint information for a given role. For a
deployment with DB2 you can have more than one endpoint, for example, an
endpoint for application developer one for database administrator.

6. Click **Logs** to view the log information. For more information, see “Viewing
virtual application instance logs” on page 626.

7. Click **Manage** in the top right for more advanced monitoring details and
operations. The Virtual Machine Monitoring page displays. If you want to
return to the **Maintain** view, click **Maintain** in the top right. For more
information about monitoring, see “Monitoring virtual application instances”
on page 626.

**Results**

Your virtual application instance is successfully deployed and started. To stop the
virtual application instance, select the application from the list, and click **Stop**.
However, a stopped deployment cannot be restarted.

To redeploy a virtual application, select the virtual application from the Virtual
Application Patterns palette, and click the **Deploy** icon in the Pattern Builder.

To remove a stopped application, select it from the Virtual Application Patterns
palette, and click **Delete**.

**What to do next**

After you deploy your virtual application, you can use the IP address of the virtual
machines to access the application artifacts. For example:

To gain access to your virtual machine after deployment type:
```
ssh -i id_rsa.txt virtuser@<your_workload_ip>
```

For SCP:
```
scp -i id_rsa.txt myfiles.txt virtuser@<your_workload_ip>
```

You can now log in to your virtual machine without a password.

To gain root access:
```
sudo su -
```

To run a command with root access:
```
sudo /sbin/ifconfig
```

You can view and monitor statistics for your deployed virtual machines and
download and view the log files from the user interface. For more information, see
“Monitoring virtual application instances” on page 626.
Monitoring virtual application instances

Use reports and charting to monitor the status and performance of your deployed virtual application instances and machines.

Before you begin

Your applications must be deployed and all of your virtual machines started before you can monitor results.

The user who deployed the application, along with those with access or the administrator, can see the monitoring information for the deployment.

About this task

Use the user interface to monitor the following statistics for your deployed virtual machines.

- Memory
- Network
- Process
- Storage

Use the user interface to monitor middleware:

- Roles

Procedure

1. Click PATTERNS > Instances > Virtual Application Instances. The Virtual Application Instances palette displays.
2. Select a virtual_application_instance. The virtual application instance details display to the right.
3. Click the More information and advanced operations icon located in the upper right corner.

Results

You have viewed and monitored virtual application instances and machines.

Viewing virtual application instance logs:

You can view logs of the virtual application instances.

Before you begin

Your virtual application patterns must be deployed and all of your virtual machines started before you can monitor results.

Procedure

1. Click PATTERNS > Instances > Virtual Application Instances. The Virtual Application Instances palette displays with the virtual application instances listed.
2. Select a virtual_application_instance. The details display to the right.
3. Click **Log**, located under the VM Status column to view the virtual machine status logs. The Log Viewer palette displays with organized log sections, such as operating system log, pattern type plug-in log and agent log.

The following types of logs can be viewed in the Log Viewer:

- **Lightweight Directory Access Protocol (LDAP) logs:**
  - `/home/idsldap/sqllib/log`
  - `db2dump`
  - `db2dump/events`
  - `db2dump/stmmlog` files
  - `/home/idsldap/idsslapd-idsldap/etc` and logs
  - `/var/idsldap/V6.3` log files

- **WebSphere Application Server (WAS) logs:**
  - `logs/server1` files
  - `logs/ffdc` files

**Attention:** If you have specified additional log files or directories to monitor in the Logging Policy, these logs also display in this list.

The log viewer is used to view the trailing 10 lines of the log you selected. New log entries are appended into the log viewer as they occur. The log viewer has several actions that control the behavior of the log viewer.

You can specify a string to filter what files are displayed in the log viewer. Strings can be prefixed by a tag that specifies one or more elements of the logs to be examined. The following tags are supported: role, dir, vm, and file.

**role:**DB2

Specifies files that belong to a role with a name that contains "DB2".

**dir:**var

Specifies files with an absolute path that includes a directory name that contains "var". The dir: tag applies to the entire path, with the exception of the file name.

**vm:**application

Specifies files on a virtual machine with a name that contains "application".

**file:**trace

Specifies files that have names that contain the word "trace".

If a tag is not included in a search string, the filter is assumed to have the file: tag.

Multiple tags can be used in conjunction.

**role:**DB2,trace

Specifies files with a name that contains "trace", and that belongs to a role with a name that contains "DB2".

Ensure that the tags: "role", "dir", "vm", and "file" are lowercase, or they are not recognized as tags. The strings following the tags, are case insensitive. For example, the string "role:WAS" will match a role name "WAS" as well as a role named "was", but the string "ROLE:was" does not match anything, because ROLE is not recognized as a valid tag. After entering a string in the filter box, click **Go** to apply the filter to the tree.

4. Click **Download All** to download all files in the log viewer. It is only displayed when no filter string has been entered, or when the filter input box has been cleared. Clicking **Download All** returns an archived, compressed file containing all of the logs on the virtual machine. If there are multiple virtual machines...
displayed in the log viewer, a separate archive file will be returned for each virtual machine. When you enter a string into the filter box to select a subset of logs, Download All is replaced by Download Filtered. Clicking Download Filtered downloads all of the files that are displayed in the filtered tree as a single archive file.

5. You can also view the log files in the Virtual Application Console. After you select the virtual application instance and the details display to the right, you can click the More information and advanced operations icon located in the upper right corner. Select the virtual machine for which you want to view logs. Click Logging on the dashboard.

6. Expand each section to view the logs.

7. Optional: Download the log file. After you expand the log type and select a log, you can click the green arrow to download the log.

Results

You have viewed the logs associated the virtual application instances and the virtual machines that they run on.

Working with shared services

Shared services provide a predefined virtual application pattern that is deployed and shared by multiple application deployments, such as virtual applications and virtual systems in the cloud.

To view shared services, you must be assigned the admin role.

A shared service provides certain runtime services to multiple applications or services to the end user on behalf of multiple applications. Only one instance of a type of shared service can be deployed in a cloud group. This shared service can be used by all application deployments in the cloud group.

Shared services provide the following features:

- A plug-in design service providing full lifecycle management capabilities in the same way as a virtual application.
- Specific runtime services to multiple applications, or services to end-users on behalf of multiple applications
- A simplified consumer (users/application deployments) and provider (implementation/Shared Service deployment) model.
- Viewed as a multi-tenant service.

Several shared services are offered by default and can be deployed as needed.

Deploying shared services

Use the IBM Cloud Orchestrator user interface to deploy and manage shared services.

Before you begin

To view shared services, you must be assigned the admin role.
About this task

Only one instance of a type of shared service can be deployed in a cloud group. This shared service can be used by all application deployments in the cloud group.

To deploy a shared service, complete the following steps:

Procedure
1. Click PATTERNS > Deployer Configuration > Shared Services.
2. In the left pane of the Shared Services window, select the shared service that you want to deploy.
3. Click the Deploy icon. The Configure and deploy application window is displayed.
4. Provide information into the fields to configure the shared service. The information that you must provide differs depending on the shared service that you are working with.
5. Click OK. The Deploy Pattern window is displayed.
6. Complete the following fields to deploy the application as a shared service:
   
   **Name**  
   Specifies that name of the application that is being shared as a service. Do not use more than two consecutive underscore characters in a shared service name.

   **Target cloud group**  
   Provides information related to cloud groups.

   **SSH Key**  
   If you want to upload a public key so that you can connect to the deployed virtual machines using SSH, click Advanced options and complete the SSH section. If you do not have an existing SSH key pair, you can generate one that can be reused with other deployments by clicking Generate. The SSH Key field populates with the generated public key. Select Download or Download (PKCS1 format) to save the private key to your local system.

   **Schedule deployment**  
   Specifies the start and end dates for the deployment.

7. To stop the shared service, click the Stop the selected shared service icon. The shared service status is displayed as STOPPED in the details.

Monitoring shared services

Use the IBM Cloud Orchestrator user interface to monitor the shared service instances that you deployed in your environment.

Complete the following steps:
1. Click PATTERNS > Instances > Shared Service Instances.
2. Select the shared service that you want to monitor.

Important: Ability to view instances of shared services in the user interface requires that users have the admin role. Additionally, access to each shared service instance must be granted to each individual user.
**Monitoring - Application**

The Monitoring - Application shared service can be deployed to one or more cloud groups to provide a reference to an external Tivoli Monitoring installation Version 6.2.2 Fix Pack 5 or later. Once created, the UNIX or Linux OS monitoring agents and the Workload monitoring agent that are provided in the virtual application workloads are automatically connected to a defined instance of a Tivoli server by using the supplied primary and fail-over Tivoli Enterprise Monitoring Server, protocol, and port. The URL for the Tivoli Enterprise Portal Webstart console is provided, so cloud administrators are presented with a monitoring link in the console to launch to the Tivoli Enterprise Console.

You must install the latest Application Support and Language Pack files for the Workload monitoring agent on the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server before creating the shared service and deploying patterns so that Tivoli Monitoring displays the new agents.

**Deploying a monitoring shared service**

Use the user interface to deploy a monitoring shared service to the system. The monitoring service is shipped with the product in the foundation pattern type. This process includes adding the shared service and deploying the shared service into the cloud system to create a shared services instance.

**Procedure**

1. Click PATTERNS > Deployer Configuration > Shared Services.
2. In the left pane of the Shared Services window, select the Monitoring-Application service.
3. Click the Deploy icon. The Configure and deploy application window is displayed.
4. Complete the following information:
   a. Enter the primary server in the Primary Server field.
   b. Enter the secondary server in the Secondary Server field.
   c. Select the protocol radio button in the Protocol field. Choose IP.PIPE, IP.SPIPE, or IP.UDP.
   d. Enter the port number in the Port field.
   e. Enter the console URL in the Console URL field.
5. Click OK. The Deploy Virtual Application window is displayed.
6. Select the target cloud group.
7. Click Advanced options to set up SSH access. The SSH key provides access to the virtual machines in the cloud group for troubleshooting and maintenance purposes.
   a. Use one of the following options to set the public key:
      - To generate a key automatically, click Generate.
      - To use an existing SSH public key, open the public key file in a text editor and copy and paste it into the SSH Key field.

      **Note:** Do not use `cat`, `less`, or `more` to copy and paste from a command shell. The copy and paste operation adds spaces to the key which prevent you from accessing the virtual machine.
   b. If you generated a key automatically, click Download to save the private key file to a secure location. The default name is `id_rsa.txt`. 
The system does not keep a copy of the private key. If you do not download the private key, you cannot access the virtual machine, unless you generate a new key pair. You can also copy and paste the public key, save the key, and reuse the same key pair for another deployment. When you have the private key, make sure that it has the correct permissions (chmod 0400 id_rsa.txt). By default, the SSH client does not use a private key file that provides open permission for all users.

8. Click OK.

Results
You deployed a monitoring shared service as an application.

Enabling TOSCA support for IBM Cloud Orchestrator

IBM Cloud Orchestrator supports importing, deploying, and exporting service templates according to the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA). This enables the consumption of third-party content provided in a standardized format.

About this task
You can enable TOSCA support by installing and enabling the pattern type “TOSCA Foundation Pattern Type”. To enable TOSCA support in your environment, you must first enable this pattern type.

Procedure
1. Get the /utils/TOSCA/tosca.foundation-1.1.0.0.tgz file from the installation package.
2. Log in to the Self-service user interface as a Cloud Administrator.
3. Click PATTERNS > Deployer Configuration > Pattern Types.
4. Click New.
5. Choose the tosca.foundation-1.1.0.0.tgz file from your local drive and click OK. A new entry is displayed in the palette.
6. Find the TOSCA Foundation Pattern Type entry in the list on the left. This entry is disabled by default.
7. Click the entry and select Enable from the Status field.

Results
After the pattern type is enabled, you can import TOSCA Cloud Service Archives (CSARs) that contain service templates that comply with the TOSCA specification. You can import the templates as Virtual Application Patterns by navigating to PATTERNS > Pattern Design > Virtual Application Patterns, or as Virtual Application Templates by navigating to PATTERNS > Pattern Design > Virtual Application Templates.

Remember: In IBM Cloud Orchestrator, the TOSCA service templates are imported and deployed as virtual application patterns or virtual application templates. Before you deploy an application pattern imported from a TOSCA CSAR, you must perform the basic configuration for virtual application patterns, such as configuration of cloud groups or import of base images.
Configuring the RPM repository for the deployment of TOSCA patterns

Before you can deploy any application patterns imported from TOSCA CSARs, you must first configure an RPM repository. With the configured RPM repository, you can install additional RPM packages into the base operating system image that is used to deploy virtual application patterns.

About this task

You must configure an RPM repository because virtual machines deployed into private clouds typically do not have access to the internet so default RPM repositories available on the internet cannot be used.

The TOSCA foundation pattern type includes a plug-in that enables the use of a dedicated RPM repository. This repository must be hosted on a network that is accessible by deployed virtual machines. The plug-in also provides configuration options that you can use to specify all the parameters of an RPM repository.

Procedure

1. Log on to IBM Cloud Orchestrator user interface as administrator.
2. Click PATTERNS > Deployer Configuration > System Plug-ins.
3. From the list on the left, select TOSCA Foundation Pattern Type 1.1.0.0 to define the pattern type.
4. From the list on the left, select the tosca.rpmrepository 1.1.0.0 plug-in.
5. From the menu on the right, click Configure. The plug-in configuration window opens.
6. Choose the type of repository to configure. You can choose the types of repositories to configure or you can select No Repository to disable the configuration of a previously created repository. You can configure the following types of repositories:
   - ISO Image on NFS
   - FTP
   - HTTP

Configuring an RPM repository of type ISO Image on NFS

This type of repository refers to an ISO image, typically the image of the operating system installation DVD, that is located on an NFS server which is accessible by deployed virtual machines.

Before you begin

Perform the steps described in “Configuring the RPM repository for the deployment of TOSCA patterns.”

Procedure

1. From the Repository Type list, select the option ISO Image on NFS.
2. Provide the following parameters:

   Repository Name
   This parameter specifies the name of the repository that is configured in the operating system of deployed virtual machines, for example RHEL_6.2_Repo.
Mount point for ISO Image
This parameter specifies a mount point that is created for mounting the ISO image inside the operating system of deployed virtual machines.

Host name or IP address
This parameter specifies the fully-qualified host name or IP address of the NFS server that hosts the ISO image. This server must be accessible by deployed virtual machines.

Repository Path
This parameter specifies the full path of the ISO image file on the NFS server, including the image file name.

Configuring an RPM repository of type FTP
This type of repository refers to an RPM repository that is available by way of an FTP server which is accessible by deployed virtual machines.

Before you begin
Perform the steps described in “Configuring the RPM repository for the deployment of TOSCA patterns” on page 632.

Procedure
1. From the Repository Type list, select the option FTP.
2. Provide the following parameters:
   - Repository Name
     This parameter specifies the name of the repository that is configured in the operating system of deployed virtual machines, for example RHEL_6.2_Repo.
   - Mount point for ISO Image
     This parameter does not have any meaning for the FTP repository type.
   - Host name or IP address
     This parameter specifies the fully-qualified host name or IP address of the FTP server that hosts the repository. This server must be accessible by deployed virtual machines.
   - Repository Path
     This parameter specifies the full path of the repository on the FTP server.

Configuring an RPM repository of type HTTP
This type of repository refers to an RPM repository that is available by way of an HTTP server which is accessible by deployed virtual machines.

Before you begin
Perform the steps described in “Configuring the RPM repository for the deployment of TOSCA patterns” on page 632.

Procedure
1. From the Repository Type list, select the option HTTP.
2. Provide the following parameters:
   - Repository Name
     This parameter specifies the name of the repository that is configured in the operating system of deployed virtual machines, for example RHEL_6.2_Repo.
**Mount point for ISO Image**
This parameter does not have any meaning for the HTTP repository type.

**Host name or IP address**
This parameter specifies the fully-qualified host name or IP address of the HTTP server that hosts the repository. This server must be accessible by deployed virtual machines.

**Repository Path**
This parameter specifies the full path of the repository on the HTTP server.

### Disabling the configuration of an RPM repository
Disable the configuration of an RPM repository to revert to the default configuration.

**Procedure**
1. Perform the steps described in “Configuring the RPM repository for the deployment of TOSCA patterns” on page 632.
2. From the Repository Type list, select the option No Repository.

**Results**
By selecting this option, you disable the configuration of an RPM repository and you keep the default configuration defined in the base operating system image used for virtual application deployment.

**Example**
You can use this option if the deployed virtual machines have access to default repositories on the internet or if the base operating system image is already configured to use specific RPM repositories.

### Configuring Chef support for TOSCA based virtual application patterns
You can configure the use of Chef automation in context of TOSCA patterns. The TOSCA Chef support inside IBM Cloud Orchestrator is a plug-in as part of the TOSCA Foundation Pattern Type.

**About this task**
For maximum flexibility, you can configure a Chef client for use with the TOSCA Chef plug-in, in the required version. Chef client is not shipped with IBM Cloud Orchestrator. To configure the Chef client for use with virtual application patterns, complete the following steps.

**Procedure**
1. To use Chef with virtual application patterns, you must upload an RPM containing a Chef client. Retrieve the Chef client RPM from Opscode.
2. Configure the TOSCA Chef Plug-in repository:
   a. Go to Patterns > System Plugins and choose the tosca.chef plug-in.
   b. Click Configure in the upper right corner. The configuration options for the TOSCA Chef plug-in appears.
   c. Choose Chef Deployment Mode: “Chef Solo”, as TOSCA service templates, imported as virtual application patterns, are using Chef Solo deployments.
d. In the Chef client package, click Browse and choose the Chef RPM that you downloaded from Opscode to your local drive.

e. Click Update to activate your changes.

The TOSCA service template, imported as virtual application pattern, uses this RPM package for installing Chef Solo on the deployed VM and executing the Chef artifacts.
Chapter 8. Integrating

Learn how to integrate IBM Cloud Orchestrator with the following IBM products.

Integrating with IBM Tivoli Monitoring

To integrate IBM Cloud Orchestrator with IBM Tivoli Monitoring, you must prepare a base operating system, set up the required databases, install the Tivoli Monitoring components, and deploy the Monitoring Agents to monitor the IBM Cloud Orchestrator environment.

Preparing a base operating system

IBM Tivoli Monitoring 6.3.0 supports several operating systems, but the IBM Cloud Orchestrator solution is based on Red Hat Enterprise Linux (RHEL), which makes it an optimal system for setting up Tivoli Monitoring.

Before you begin

For a list of supported operating systems, see Supported operating systems.

For more information about hardware and software requirements for IBM Tivoli Monitoring, see Hardware and software requirements.

Procedure

1. You must install several rpm packages that are required by IBM Global Security Toolkit (GSKit). GSKit is deployed automatically with the Tivoli Monitoring installation and requires the following operating system patches:
   - ksh-20091224-1.el6.x86_64.rpm
   - glibc-2.12-1.7.el6.i686.rpm
   - libgcc-4.4.4-13.el6.i686.rpm
   - nss-softokn-freebl-3.12.7-1.1.el6.i686.rpm
2. Install the libraries that are required by the OS Monitoring Agent:
   - libstdc++
   - libgcc
   - compat-libstdc++

Restriction: On a 64-bit system, you must have 32-bit and 64-bit versions of those libraries.
**Database setup**

IBM Tivoli Monitoring requires two databases, the Tivoli Enterprise Portal Server database and the Tivoli Data Warehouse database.

- The Tivoli Enterprise Portal Server database, or portal server database, stores user data and information that is required for graphical presentation on the user interface. The portal server database is created automatically during configuration of the portal server. It is always on the same computer as the portal server.

- The Tivoli Data Warehouse database, also called the warehouse database or data warehouse, stores historical data for presentation in historical data views. In a single-computer installation, the warehouse database is created on the same relational database management server that is used for the portal server database. In larger environments, it is best to create the warehouse database on a different computer from the portal server.

You can create a TEPS database on an embedded Derby database that is delivered with the Tivoli Monitoring installer. Warehouse database can be set on a DB2 or Oracle server. Thus, the best solution is to install a DB2 server on a Tivoli Monitoring server and use it for TEPS and Warehouse databases.

For more information about installing DB2, see the DB2 documentation.

**Installing IBM Tivoli Monitoring**

The installation of IBM Tivoli Monitoring requires several mandatory components. You can also install extra ones if you are planning to set up a dashboard environment or use products that support integration using OSLC.

**About this task**

For more information about Tivoli Monitoring and its components, see [Components of the monitoring architecture](#).

For more information about installing and configuring Tivoli Monitoring, see [High-level installation steps](#).

**Procedure**

1. You must install the following components of Tivoli Monitoring:
   - Hub Tivoli Enterprise Monitoring Server
   - Tivoli Enterprise Portal Server
   - Tivoli Enterprise Portal desktop client
   - The Warehouse Proxy Agent
   - The Summarization and Pruning Agent

2. If you plan to set up a dashboard environment, you can install extra components. For base installation, these features are not required and can be skipped or installed later:
   - Dashboard Application Services Hub (a Jazz for Service Management component)
   - IBM Infrastructure Management Dashboards for Servers
   - Tivoli Authorization Policy Server
   - tivcmd Command Line Interface for Authorization Policy
Dashboard and JazzSM are new components of Tivoli Monitoring 6.3.0.1. Dashboard does not completely replace Tivoli Portal Client but is used to improve data presentation. TEPS is used for configuration issues.

3. If you plan to use the Performance Monitoring service provider to integrate with the Jazz for Service Management Registry Services component and other products that support integration using OSLC, install the following component. For base installation, this feature is not required and can be skipped or installed later:
   - Tivoli Enterprise Monitoring Automation Server

**Packages used for installation**

You need several packages to install IBM Tivoli Monitoring 6.3.0. All of its components can also be installed in silent mode.

The following packages are required to install IBM Tivoli Monitoring 6.3.0.1:
- CIL2AEN - IBM Tivoli Monitoring V6.3.0.1 Base, Linux (64-bit Env.), English
- CIL2HML - IBM Tivoli Monitoring V6.3.0.1 Dashboards for Servers and Authorization Policy Components Assembly Multiplatform, Multilingual
- CIGL8ML - IBM Tivoli Monitoring V6.3.0.1 Language Support

All IBM Tivoli Monitoring components can be installed and configured in silent mode. First, you must install all products with one silent_install file and then configure each one of them with silent_config response files. For more information about modifying the files, see the following examples.

- IBM Tivoli Monitoring: modify the silent_install.txt file with the following information:
  
  ```
  INSTALL_PRODUCT=ms
  INSTALL_PRODUCT=cq
  INSTALL_PRODUCT=hd
  INSTALL_PRODUCT=sy
  INSTALL_PRODUCT_TMS=all
  INSTALL_PRODUCT_TPS=all
  INSTALL_PRODUCT_TPW=all
  INSTALL_ENCRYPTION_KEY=IBMTivoliMonitoringEncryptionKey
  SEED_TEMS_SUPPORTS=true
  MS_CMS_NAME=TEMS
  DEFAULT_DISTRIBUTION_LIST=NEW
  ```

- Tivoli Enterprise Monitoring Server: modify the ms_silent_config.txt file with the following information:
  ```
  HOSTNAME=itmsrv1
  NETWORKPROTOCOL=ip.pipe
  SECURITY=YES
  ```

- Tivoli Enterprise Portal Server: modify the cq_silent_config.txt file with the following information:
  ```
  CMSCONNECT=YES
  HOSTNAME=itmsrv1
  NETWORKPROTOCOL=ip.pipe
  DB2INSTANCE=db2inst1
  DB2ID=itmuser
  DB2PW=passw0rd
  WAREHOUSEID=itmuser
  WAREHOUSEDB=WAREHOUSE
  WAREHOUSEPW=passw0rd
  ADMINISTRATORID=db2inst1
  ADMINISTRATORPW=passw0rd
  ```

- Summarization and Pruning Agent: modify the sy_silent_config.txt file with the following information:
Warehouse Agent: modify the `hd_silent_config.txt` file with the following information:

```plaintext
CMSCONNECT=YES
HOSTNAME=itmsrv1
NETWORKPROTOCOL=ip.pipe
KSY_WAREHOUSE_TYPE=DB2
KSY_WAREHOUSE_JARS=/opt/ibm/db2/v10.1/java/db2jcc.jar,/opt/ibm/db2/v10.1/java/db2jcc_license_cu.jar
KSY_DB2_JDBCURL=jdbc:db2://db2srv1:50001/WAREHOUSE
KSY_DB2_JDBCDRIVER=com.ibm.db2.jcc.DB2Driver
KSY_WAREHOUSE_USER=itmuser
KSY_DB_COMPRESSION=N
KSY_TIMEZONE_IND=AGENT
KSY_START_OF_WEEK_DAY=0
KSY_SHIFTS_ENABLED=N
KSY_SHIFT1_HOURS=0,1,2,3,4,5,6,7,8,18,19,20,21,22,23
KSY_SHIFT2_HOURS=9,10,11,12,13,14,15,16,17
KSY_VACATIONS_ENABLED=N
KSY_WEEKENDS_AS_VACATIONS=N
KSY_VACATION_DAYS=
SY_MAX_ROWS_PER_TRANSACTION=1000
KSY_FIXED_SCHEDULE=Y
KSY_EVERY_N_DAYS=1
KSY_HOUR_TO_RUN=2
KSY_HOUR_AM_PM=AM
KSY_MINUTE_TO_RUN=0
KSY_EVERY_N_MINS=60
KSY_BATCH_MODE=0
KSY_CNP_SERVER_HOST=localhost
KSY_CNP_SERVER_PORT=1920
KSY_HOUR_AGE_UNITS=1
KSY_DAY_AGE_UNITS=0
KSY_MAX_WORKER_THREADS=2
KSY_CACHE_MINS=10
```

Creating a warehouse database

IBM Tivoli Monitoring supports a remote data warehouse through aliases in a local DB2 server.

About this task

For more information about creating a warehouse database, see [Creating the Tivoli Data Warehouse database](#).

Procedure

1. Create a warehouse database on a remote server.
2. Create a DB2 user on a remote server. Grant the user administrative rights to the database.
3. On local DB2 on which IBM Tivoli Monitoring is installed, catalog a remote data warehouse.
Monitoring Agent for Linux

To monitor the entire IBM Cloud Orchestrator environment, install Monitoring Agent on each Linux computer and configure it with the host name of your Tivoli Enterprise Monitoring Server.

For more information about installing the operating system agents, see Installing monitoring agents.

The operating system agents are delivered with the following package:
- CIGM1EN - IBM Tivoli Monitoring V6.3.0 Agents, Multiplatform, English

If you want to use silent mode to install and configure the agents, you can use the following files:
- silent_install.txt. You can use this file without any changes.
- lz_silent_config.txt. Modify the file with the following information:
  
  CMSCONNECT=YES
  HOSTNAME=itmsrv1
  NETWORKPROTOCOL=ip.pipe

If you want the agents to report to your main Tivoli Enterprise Monitoring Server, configure each one of them with such a configuration file.

Monitoring Agent for Kernel-based virtual machines

The KVM agent is used to monitor the Region Server and it must be installed with other ITM 6.3.0 components. The agent requires the libvirt library that is used to connect with the monitored KVM hypervisor.

For more information about installing and configuring the agent, see Linux Kernel-based virtual machines agent.

To properly configure the agent, you must provide parameters that describe the hypervisor.

You must add the RSA public keys of host on which the KVM agent is deployed to the hypervisor to enable communication through the SSH protocol. The protocol is configured and enabled for the Kernel services that are created by the Region Server, but you must enable it between the Region Server and the computer on which IBM Tivoli Monitoring is installed.

For more information about configuring the SSH protocol, see SSH protocol.

The KVM agent requires the following packages:
- CIGN0EN - IBM Tivoli Monitoring for Virtual Environments V7.2 VMware VI, KVM, NetApp Storage and NMA Agents and Support files (TMVE), Windows and Linux, English, Multiplatform
- CIGN2ML - IBM Tivoli Monitoring for Virtual Environments V7.2 Agent Language Pack (TMVE), Multiplatform, Multilingual

If you want to use silent mode for installing and configuring the operating system agents, you can use the following files:
- Modify the silent_install.txt file with the following information:
  
  INSTALL_PRODUCT=v1
  INSTALL_PRODUCT_TMS=all
  INSTALL_PRODUCT_TPS=all
Modify the v1_silent_config.txt with the following information:

- CMSCONNECT=YES
- HOSTNAME=itmsrv1
- NETWORKPROTOCOL=ip.pipe
- INSTANCE=RegionServer
- DATA_PROVIDER.KV1_LOG_FILE_MAX_COUNT=10
- DATA_PROVIDER.KV1_LOG_FILE_MAX_SIZE=5190
- DATA_PROVIDER.KV1_LOG_LEVEL=INFO
- HOST_ADDRESS.RegionServer=<host name of the region server visible by itmsrv1>
- USERNAME.RegionServer=root
- PROTOCOL.RegionServer=ssh
- PORT.RegionServer=22
- CONNECTION_MODE.RegionServer=system

**OpenStack hypervisors**

With the default configuration, you can monitor the Region Server as a KVM hypervisor. To do so, you can use Monitoring Agent for Kernel-based virtual machines from Tivoli Monitoring for Virtual Environments.

OpenStack can use different hypervisors, like KVM or VMware. To monitor different KVM hypervisors, you can use the installed agent and simply add a new instance in the agent configuration.

To monitor a VMware hypervisor, you must install and configure Monitoring Agent for VMware, which is also included in Tivoli Monitoring for Virtual Environments. Then, you can add an instance of configuration every time a new hypervisor is added to OpenStack.

For more information about VMware Agent, see [VMware VI User’s Guide](#).

**Integrating with IBM Endpoint Manager**

To integrate IBM Cloud Orchestrator with IBM Endpoint Manager follow these topics.

**About this task**

The purpose of this document is to demonstrate how IBM Endpoint Manager can be integrated with IBM Cloud Orchestrator. The first topic describes a tool which can be used to deploy Endpoint Manager agents to existing computers. The following topic describes how a script package pattern component can be created to automatically install the IBM Endpoint Manager agent when deploying a virtual system. Installing the agent as a script package eliminates the need to manually install agents on newly deployed virtual systems. The procedure described in that topic also demonstrates policy-based patch management of virtual systems provisioned through IBM Cloud Orchestrator.

**Note:** Install and set up IBM DB2 V10.5 to support IBM Cloud Orchestrator. If you subsequently install IBM Endpoint Manager, do not install the IBM DB2 Workgroup Server Edition that is included with IBM Endpoint Manager. Continue to use the IBM DB2 Enterprise Server Edition V10.5, because IBM Endpoint Manager will be used in the context of IBM Cloud Orchestrator.
Installing Endpoint Manager agent to existing computers

Using the IBM Endpoint Manager agent deployment wizard, you can deploy the IBM Endpoint Manager agent to computers in your network.

Use the wizard to enter the details of the computers to which you want to deploy the agent and then select the agent version and the client configuration. The wizard provides feedback on each step of the process and displays the results of the deployment for each individual computer.

Download the Beta version of the Endpoint Manager agent deployment wizard at: http://software.bigfix.com/download/bes/util/TEMAgentDeployment-1.0.282-win.zip.

Installing Endpoint Manager agents during virtual machine deployment

Use a custom script package to deploy Endpoint Manager agents during virtual machine deployment.

To integrate Endpoint Manager with the IBM Cloud Orchestrator, create a custom script package. When the pattern associated with the script package is deployed an Endpoint Manager agent, it is installed on the target system. The agent allows registration and communication between the target system and the Endpoint Manager server.

The behavior of parts in IBM Cloud Orchestrator topologies can be customized by adding script packages to pattern topologies. Custom script packages can be created and then added to the required part within the pattern containing that part.

Policy-based patch management of virtual systems can be achieved by configuring groups, within Endpoint Manager, based on the value of a specific environment variable set on the managed virtual machine. The value of this specific environment variable can be defined during the deployment of the virtual machine, in other words, during pattern deployment in IBM Cloud Orchestrator.

Endpoint Manager install agent script package

A script package is a compressed file that contains the executable file and all associated files for the script package. The compressed file includes the executable file (script.sh on Linux, or script.bat or script.cmd on Windows) plus the .json file (cbscript.json) needed to run the script on the deployed virtual machine.

The Endpoint Manager install agent script package contains the following files:

- Endpoint Manager Server Masthead file (actionsite.afxm).
- Endpoint Manager agent installer (*.rpm / setup.exe).
- Script package executable (script.sh or script.bat).
- JSON configuration file (cbscript.json).

where:

Endpoint Manager Server Masthead file

The masthead file is generated during installation of the target Endpoint Manager server. The masthead file contains configuration, license, and security information unique to each installation. The file has the extension .afxm and can be retrieved in the following ways:
In the database of the target IBM Tivoli Endpoint Manager server. Use the Admin Tool to export the masthead file.

On the target IBM Endpoint Manager server named Actionsite.afxm, and found in one of the following site version folders located at: C:\Program Files\BigFix\Enterprise\BES\Server\sitearchive\actionsite\archiveDir\actionsite_<XXX>

Note: If the masthead is not named actionsite.afxm, rename it to actionsite.afxm.

**Endpoint Manager Agent Installer**

This file is the installer for the Endpoint Manager agent and is dependent on the Endpoint Manager server version and on the operating system version the agent is being deployed to. It can be retrieved in the following ways:

**Windows:**
The Endpoint Manager agent installer can be found on the Endpoint Manager server in the installation folder: C:\Program Files (x86)\BigFix\Enterprise\BES Console\BESClientDeploy\BigFixInstallSource\ClientInstaller. The file is named setup.exe.

**Linux:** The Endpoint Manager agent installer for non-windows operating systems can be found at [http://support.bigfix.com/install/besclients-nonwindows.html](http://support.bigfix.com/install/besclients-nonwindows.html). The file is named BESAgent<Endpoint Manager version>-<os version>.rpm. For example, on Red Hat Linux: BESAgent-8.2.1175.0-sle11.x86_64.rpm.

**Script package executable**

This file is created manually. It is executed when the script package is run on the newly deployed system. It configures and runs the Endpoint Management agent installer file:

**Windows:**

script.bat

**Linux:** script.sh

**JSON configuration file**

This file is created manually. It is used to populate all the information needed to configure a script package when it is added to the catalog. The file must be called cbscript.json.

**Policy-based Patch Management**

Policy-based patch management can be achieved using groups in Endpoint Manager. To configure groups of virtual systems within Endpoint Manager, create a Computer Group which groups computers based on the value of the _BESCLIENT_GROUP_NAME environment variable.

For example, in an Automatic Group, set the property **Relevance Expression** to true and add the following constraint:

exists setting "_BESClient_GROUP_NAME" of client AND
value of setting "_BESClient_GROUP_NAME" of client = "<MY_VALUE>".

This groups any computers which have the _BESCLIENT_GROUP_NAME set to <MY_VALUE>. During the deployment of the virtual system, _BESCLIENT_GROUP_NAME should be set to MY_VALUE to be included in this group.
To assign a virtual machine to a specific group during deployment, the Endpoint Manager install agent script package contains the following environment variable: _BESCLIENT_GROUP_NAME_. This environment variable corresponds to the _BESCLIENT_GROUP_NAME_ client setting configured within the Endpoint Manager server and can be used to group virtual systems within the Endpoint Manager server.

The _BESCLIENT_GROUP_NAME_ environment variable is by default set to none and ignored but can be modified when deploying a virtual system.

**Example script package to install an Endpoint Manager agent on Red Hat Linux**

This script package installs an Endpoint Manager agent on a Red Hat Linux system and configures an environment variable which is used by the Endpoint Manager server to group virtual machines:

- Retrieve the Endpoint Manager Server Masthead from the Endpoint Manager Server machine. If the masthead is not named actionsite.afxm, rename it to actionsite.afxm.
- Retrieve the appropriate Red Hat Linux agent installer (.rpm) from [http://support.bigfix.com/install/besclients-nonwindows.html](http://support.bigfix.com/install/besclients-nonwindows.html), for example, BESAgent-8.5.6666.0-rhe5.x86_64.rpm.
- Create the install.sh script as follows, replacing the `<SERVER IP>` and `<SERVER HOSTNAME>` placeholders with the correct IP and host name values corresponding to the Endpoint Manager server machine. If the Endpoint Manager server port is not the default value of 52311, replace this value with the correct port number:

```bash
# Source the env file
if [ -f "/etc/virtualimage.properties" ]; then.
    /etc/virtualimage.properties
fi

# install the IEM client package
rpm -ivf /tmp/IEMClient/BESAgent*.rpm

# copy the actionsite masthead
cp /tmp/IEMClient/actionsite.afxm /etc/opt/BESClient

# resolve IEM server shortname to ip address
echo "<SERVER IP>"<SERVER HOSTNAME>">/etc/hosts

# Add the IEM Server port 52311 to iptables if
if [ -f "/etc/init.d/SuSEfirewall2_init" ]; then
    # SUSE/SLES host
    /etc/init.d/SuSEfirewall2_init status /dev/null 2>&1
    if [ $? -eq 0 ]; then
        /etc/init.d/SuSEfirewall2_setup stop
        iptables -A INPUT -p tcp --dport 52311 -j ACCEPT
        iptables -A OUTPUT -p tcp --dport 52311 -j ACCEPT
        /etc/init.d/SuSEfirewall2_setup reload
    fi
else
    /etc/init.d/iptables status >/dev/null 2>&1
    if [ $? -eq 0 ]; then
        # save the iptables and then stop it
        /etc/init.d/iptables save
        /etc/init.d/iptables stop
        iptables -A INPUT -p tcp --dport 52311 -j ACCEPT
        iptables -A OUTPUT -p tcp --dport 52311 -j ACCEPT
        /etc/init.d/iptables save
        /etc/init.d/iptables start
```

Chapter 8. Integrating 645
# If the _BESCLIENT_GROUP_NAME is set (default is none i.e. ignore)
if [ "${_BESCLIENT_GROUP_NAME}" == "none" ]; then
    echo "Group policy to be set to: ${_BESCLIENT_GROUP_NAME}"
    echo "[Software\BigFix\EnterpriseClient\Settings\Client\_BESCLIENT_GROUP_NAME]">>/var/opt/BESClient/besclient.config
    echo "Value = ${_BESCLIENT_GROUP_NAME}">>/var/opt/BESClient/besclient.config
fi
# start the IEM client
/etc/init.d/besclient start

• Create the cbscript.json file. Make sure that the name specified in the file matches the script package zip name, for example: IEM_8.5.6666.0-rhe5.x86_64.zip:

```
[
  {
    "name": "IEM_8.5.6666.0-rhe5.x86_64",
    "version": "1.0.0",
    "description": "This script package installs the IEM Agent on RHEL 5 64-bit ",
    "command": "/bin/sh /tmp/IEMClient/install.sh",
    "log": "/tmp/IEMClient",
    "location": "/tmp/IEMClient",
    "timeout": "0",
    "commandargs":"
    "keys":
    [
      {
        "scriptkey": "_BESCLIENT_GROUP_NAME",
        "scriptvalue": "",
        "scriptdefaultvalue": "none"
      }]
  }]
```

• The script package name must match the name specified in the cbscript.json file. Create the script package by zipping together the following files (for example, IEM_8.5.6666.0-rhe5.x86_64.zip):
  – actionsite.afxm - the Endpoint Manager Server Masthead.
  – BESAgent-8.5.6666.0-rhe5.x86_64.rpm - the agent installer for Red Hat Linux.
  – install.sh.
  – cbscript.json.

• Import the script package zip file into IBM Cloud Orchestrator and keep the default setting Executes at virtual system creation.

• Create a pattern containing a Red Hat Linux part and add the script package to this part.

• Optional. Configure a Computer Group, in the Endpoint Manager Server console, to match the proposed _BESCLIENT_GROUP_NAME environment variable value set when the virtual system is deployed.

• Deploy the pattern. Optional: Specify the value for the _BESCLIENT_GROUP_NAME environment variable to match the Computer Group configured in the Endpoint Manager console. By default this value is set to none and is ignored.

• The script package runs when the virtual system is deployed. To verify that the agent is installed correctly review the log files.

• After the agent has registered with the Endpoint Manager server, the computer system information is displayed in the Endpoint Manager console.

• If a group has been configured and set, the computer also displays under the specific group heading in the Endpoint Manager console.
• You can now perform patch management.

**Example script package to install an Endpoint Manager agent on SUSE Linux**

This script package installs an Endpoint Manager agent on a SUSE Linux Enterprise system and configures an environment variable which is used by the Endpoint Manager server to group virtual machines.

• Retrieve the Endpoint Manager Server Masthead from the Endpoint Manager Server machine. If the masthead is not named actionsite.afxm, rename it to actionsite.afxm.

• Retrieve the appropriate SUSE Linux Enterprise agent installer (.rpm) from [http://support.bigfix.com/install/besclients-nonwindows.html](http://support.bigfix.com/install/besclients-nonwindows.html) for example BESAgent-8.2.1175.0-sle11.x86_64.rpm

• Create the install.sh script as follows, replacing the <SERVER IP> and <SERVER HOSTNAME> placeholders with the correct IP and host name values corresponding to the Endpoint Management server machine. If the Endpoint Manager server port is not the default value of 52311, replace this value with the correct port number:

```bash
# Source the env file
if [-f "/etc/virtualimage.properties"]; then
  . /etc/virtualimage.properties
fi

# install the IEM client package
rpm -ivf /tmp/IEMClient/BESAgent*.rpm

# copy the actionsite masthead
cp /tmp/IEMClient/actionsite.afxm /etc/opt/BESClient

# resolve IEM server shortname to ip address
echo "<SERVER IP><SERVER HOSTNAME>>" >>/etc/hosts

# Add the IEM Server port 52311 to iptables
if [-f "/etc/init.d/SuSEfirewall2_init"]; then
  # SUSE/SLES host
  /etc/init.d/SuSEfirewall2_init status >/dev/null 2>&1
  if [ $? -eq 0 ]; then
    /etc/init.d/SuSEfirewall2_setup stop
    iptables -A INPUT -p tcp --dport 52311 -j ACCEPT
    iptables -A OUTPUT -p tcp --dport 52311 -j ACCEPT
    /etc/init.d/SuSEfirewall2_setup reload
  fi
else
  /etc/init.d/iptables status >/dev/null 2>&1
  if [ $? -eq 0 ]; then
    # save the iptables and then stop it
    /etc/init.d/iptables save
    /etc/init.d/iptables stop
    iptables -A INPUT -p tcp --dport 52311 -j ACCEPT
    iptables -A OUTPUT -p tcp --dport 52311 -j ACCEPT
    /etc/init.d/iptables save
    /etc/init.d/iptables start
  fi
fi

# If the _BESCLIENT_GROUP_NAME is set (default is none i.e. ignore)
if [ ! "${_BESCLIENT_GROUP_NAME}" == "none" ]; then
  echo "Group policy to be set to: " ${_BESCLIENT_GROUP_NAME}
  echo "[Software\BigFix\EnterpriseClient\Settings\Client\_BESCLIENT_GROUP_NAME]"
  >>/var/opt/BESClient/besclient.config
  echo "value = "${_BESCLIENT_GROUP_NAME}"" >>/var/opt/BESClient/besclient.config
```
# start the IEM client
/etc/init.d/besclient start

- Create the cbscript.json file: Make sure that the name specified in the file below matches the script package zip name, for example, IEM_8.2.1175.0-sle11.x86_64.zip:

```
{
  "name": "IEM_8.2.1175.0-sle11.x86_64",
  "version": "1.0.0",
  "description": "This script package installs the IEM Agent on SUSE 11",
  "command": "/bin/sh /tmp/IEMClient/install.sh",
  "log": "/tmp/IEMClient",
  "location": "/tmp/IEMClient",
  "timeout": "0",
  "commandargs": "",
  "keys": [
    {
      "scriptkey": "_BESCLIENT_GROUP_NAME",
      "scriptvalue": "",
      "scriptdefaultvalue": "none"
    }
  ]
}
```

- The script package name must match the name specified in the cbscript.json file. Create the script package by zipping together the following files (for example, IEM_8.2.1175.0-sle11.x86_64.zip):

  - actionsite.afxm
    The Endpoint Manager Server Masthead.

  - BESAgent-8.2.1175.0-sle11.x86_64.rpm
    The agent installer for SUSE Linux.

  - install.sh

  - cbscript.json

- Import the script package zip file into IBM Cloud Orchestrator and keep the default setting of *Executes at virtual system creation*.

- Create a pattern containing a SUSE Linux Enterprise part and add the script package to this part.

- Optional: Configure a Computer Group, in the Endpoint Manager Server console, to match the proposed _BESCLIENT_GROUP_NAME environment variable value set when the virtual system is deployed.

- Deploy the pattern. Optional: Specify the value for the _BESCLIENT_GROUP_NAME environment variable to match the Computer Group configured in the Endpoint Manager console. By default this value is set to none and is ignored.

- The script package will run when the virtual system is deployed. To verify the agent installed correctly review the log files.

- After the agent has registered with the Endpoint Manager server, the computer system information is displayed in the Endpoint Manager console.

- If a group has been configured and set, the computer is also displayed under the specific group heading in the Endpoint Manager console.

- You can now perform patch management.
Example script package to install an Endpoint Manager agent on Windows

This script package installs an Endpoint Manager agent on a Windows system and configures an environment variable which is used by the Endpoint Manager server to group virtual machines.

- Retrieve the Endpoint Manager Server Masthead from the Endpoint Manager Server machine. If the masthead is not named actionsite.afxm, rename it to actionsite.afxm.
- Retrieve the Windows installer file from the IBM Tivoli Endpoint Manager server in the C:\Program Files (x86)\BigFix Enterprise\BES Console\BESClientDeploy\BigFixInstallSource\ClientInstaller installation folder. The file is named setup.exe.
- Create the install.bat script as follows, replacing the <SERVER IP> and <SERVER HOSTNAME> placeholders with the correct IP and host name values corresponding to the Endpoint Management Server machine:

```batch
REM set the hostname
REM <SERVER IP><SERVER HOSTNAME>
set hostspath=%windir%\System32\drivers\etc\hosts
echo <SERVER IP><SERVER HOSTNAME>>%hostspath

REM navigate to the dir
cd C:\\TEMP\\TEMClient

REM check for _BESCLIENT_GROUP_NAME
IF %_BESCLIENT_GROUP_NAME% == "none"GOTO NOPATH
    :YESPATH
    IF %_BESCLIENT_GROUP_NAME% == "none"GOTO NOPATH
    REM The _BESCLIENT_GROUP_NAME environment variable was detected.
    REM create the clientsettings.cfg file.
    REM Group policy to be set to: %_BESCLIENT_GROUP_NAME%
    @ECHO _BESCLIENT_GROUP_NAME=%_BESCLIENT_GROUP_NAME%>>clientsettings.cfg
    GOTO END

:NOPATH
    REM The _BESCLIENT_GROUP_NAME environment variable was NOT detected or set to none.
    GOTO END

:END
REM install the TEM client package
setup.exe /s /v/qn
```

- Create the cbscript.json file. Make sure that the name specified in the file below matches the script package zip name, for example, IEM_8.2.1175.0-windows_setup.zip:

```json
[

{
    "name": "IEM_8.2.1175.0-windows_setup",
    "version": "1.0.0",
    "description": "This script package installs the TEM Client on WIN ",
    "command": "install.bat",
    "log": "C:\\TEMP\\TEMClient",
    "location": "C:\\TEMP\\TEMClient",
    "timeout": "0",
    "commandargs": "",
    "ostype": "windows",
    "keys":
    [
      {"scriptkey": "_BESCLIENT_GROUP_NAME",
       "scriptvalue": "",
       "scriptdefaultvalue": "none"
    ]
]
```
The script package name must match the name specified in the cbscript.json file. Create the script package by zipping together the following files (for example, IBM_8.2.1175.0-windows_setup.zip):

- actionsite.afxm
  - The Endpoint Manager Server Masthead.
- setup.exe
  - The agent installer for Windows.
- install.bat
- cbscript.json

- Import the script package zip file into IBM Cloud Orchestrator and keep the default setting of **Executes at virtual system creation**.
- Create a pattern containing a Windows operating system part and add the script package to this part.
- Optional: Configure a Computer Group, in the Endpoint Manager Server console, to match the proposed _BESCLIENT_GROUP_NAME environment variable value set when the virtual system is deployed.
- Deploy the pattern. Optional: Specify the value for the _BESCLIENT_GROUP_NAME environment variable to match the Computer Group configured in the Endpoint Manager console. By default this value is set to none and is ignored.
- The script package runs when the virtual system is deployed. To verify the agent installed correctly review the log files.
- Once the agent has registered with the Endpoint Manager server the computer system information is displayed in the Endpoint Manager console.
- If a group has been configured and set, the computer also displays under the specific group heading in the Endpoint Manager console.
- You can now perform patch management.

**Troubleshooting**

If the Endpoint Manager agent fails to register with the Endpoint Manager server, check the following issues:

- The host name where the Endpoint Manager agent is installed must be correctly resolved by using the `ping` command from the Endpoint Manager server, and vice versa. To ping the machine, use the command:
  
```
  # ping ip_address
  
  where ip_address is:
  - When located on the Endpoint Manager server, `ip_address` refers to the host where the Endpoint Manager agent is installed.
  - When located on the host where the Endpoint Manager agent is installed, `ip_address` refers to the Endpoint Manager server.
- Ensure that Port 52311 is open between the host where the Endpoint Manager is.
- Ensure that the agent is running, by executing the following command on the virtual machine:
  
```
  # /etc/init.d/besclient status
  
  If the agent is not running, start the agent by executing:
```
# /etc/init.d/besclient start

• Restart the agent within the virtual system:
  # /etc/init.d/besclient stop
  # /etc/init.d/besclient start

Note: On Windows, the client is installed as a service.

• Ensure that the host name resolves to the correct IP address. Verify the values in
  the hosts file.
```
Chapter 9. Reporting

IBM Cloud Orchestrator provides a diverse set of reports that provide specific data you can use for planning purposes.

Tivoli Common Reporting

Tivoli Common Reporting is provided for reporting of Monitoring, Metering, and Billing.

- For information about Tivoli Common Reporting, including installation, see the Jazz for Service Management information center.
- For information about using Tivoli Common Reporting for metering and billing, see the Administering reports guide in the metering and billing section.
- For information about using Tivoli Common Reporting in IBM Tivoli Monitoring, see the Tivoli Common Reporting in the IBM Tivoli Monitoring information center.
Chapter 10. Reference

The following topics provide reference information for IBM Cloud Orchestrator.

REST API reference

The representational state transfer (REST) application programming interface (API) is provided by IBM Cloud Orchestrator.

Before you begin

Each product exposes a REST API as there are no special configuration settings to enable or disable this interface. The IBM Cloud Orchestrator REST API is available on the same IP address or host name used to access the product GUI and command-line interface. Unlike the GUI, the REST API is only supported over the HTTPS protocol on port 443. The product uses a self-signed certificate for its SSL sessions. The same certificate is used for GUI, command-line interface and REST API sessions. You need to configure your HTTPS client to either accept or ignore this certificate during the SSL handshake. You must use an HTTPS client that allows you to set the HTTP headers for each request. This is because there are multiple headers required for authentication, authorization, and content negotiation.

When generating HTTP requests to the IBM Cloud Orchestrator REST API, pay special attention to the following headers:

Accept
With a few exceptions, the REST API generates JSON-encoded data in its responses. Include an "Accept: application/json" header in your request to indicate the ability of your client to handle JSON responses.

Accept-Language
Use this header on your HTTP request to specify which language or locale should be used by the product when generating the response data. You can specify any of the languages supported by the product.

Authentication
The REST API only supports HTTP basic authentication. After successfully authenticating, the server will return two cookies named zsessionid and SimpleToken that should be included with subsequent HTTP requests that are part of the same session. The same user IDs and passwords used to access the GUI and the command-line interface are used to access the REST API. The authorization of a user to perform actions on the product is independent of the interface (GUI, command-line interface or REST API) used to request the actions.

Content-Type
All the content included in an HTTP request body sent to the product must be JSON encoded. You must include a "Content-Type: application/json" header to indicate this for each request that includes any data.

X-IBM-Workload-Deployer-API-Version: 4.0.0.1
Every HTTP request to the product must include a "X-IBM-Workload-Deployer-API-Version: 4.0.0.1" header to indicate that your client expects the REST API semantics described in this document.
When not using the default domain, the HTTP request must include in the header "domainName:<yourDomainName>" to let the user be authenticated to the <yourDomainName> domain.

When not using the default project, the HTTP request must include in the header "projectName:<yourProjectName>" to let the user be authenticated in the <yourProjectName> project.

The REST API is only supports the sending and receiving of UTF-8 encoded data. Ensure that your HTTP client is appropriately set to encode and decode character data, including JSON data. All responses of REST requests in JSON format are encoded in UTF-8.

Note: Key-value pairs that are only used by user interface clients are optional.

REST API frameworks

This topic describes the REST API frameworks used in an IBM Cloud Orchestrator environment for REST calls (client and server).

They are:

- org.apache.wink 1.1.3: used by Workload Deployer and IBM Cloud Orchestrator.
- javax.net.ssl.HttpsURLConnection: used by the genericREST implemented by IBM Cloud Orchestrator that makes Rest calls from Business Process Manager to IBM Cloud Orchestrator.
- org.apache.http: used by Business Process Manager to connect to OpenStack.

For customized REST calls that are implemented outside of IBM Cloud Orchestrator, make sure that the compatible REST API framework is used. For example, if you implement a script that makes REST calls against IBM Cloud Orchestrator, you must use the org.apache.wink 1.1.3 framework or any other compatible REST API framework.

Note: For non compatible REST API framework you will get 'incompatible with ....... ' errors.

Orchestration actions REST API

You can use this set of REST API calls to interact with orchestration actions.

List all orchestration action entries

Use this REST API method to list all orchestration action entries.

Available HTTP method

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/automation?parm1=value1">https://hostname/resources/automation?parm1=value1</a> &amp;parm2=value2...</td>
</tr>
<tr>
<td>Response</td>
<td>JSON array of all orchestration action entries filtered according to the query parameters:</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>{ id:</td>
</tr>
<tr>
<td></td>
<td>name:</td>
</tr>
<tr>
<td></td>
<td>description:</td>
</tr>
<tr>
<td></td>
<td>created:</td>
</tr>
<tr>
<td></td>
<td>updated:</td>
</tr>
<tr>
<td></td>
<td>priority:</td>
</tr>
<tr>
<td></td>
<td>process:</td>
</tr>
<tr>
<td></td>
<td>process_app_id:</td>
</tr>
<tr>
<td></td>
<td>process_app_short_name:</td>
</tr>
<tr>
<td></td>
<td>process_app_name:</td>
</tr>
<tr>
<td></td>
<td>category:</td>
</tr>
<tr>
<td></td>
<td>operation_type:</td>
</tr>
<tr>
<td></td>
<td>ownerid:</td>
</tr>
<tr>
<td></td>
<td>apply_to_all_pattern:</td>
</tr>
<tr>
<td></td>
<td>event:</td>
</tr>
<tr>
<td></td>
<td>human_service:</td>
</tr>
<tr>
<td></td>
<td>human_service_app_id:</td>
</tr>
<tr>
<td></td>
<td>human_service_app_short_name:</td>
</tr>
<tr>
<td></td>
<td>human_service_app_name:</td>
</tr>
<tr>
<td></td>
<td>implementation_type:</td>
</tr>
<tr>
<td></td>
<td>icon:</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

| Return values | • 200 - OK |
|               | • 401 - Unauthorized |
|               | • 404 - Not found |

Structure of the query string:
- A parameter list is appended to the URL after the '?' mark.
- The query parameters are specified as `parameter=value`, where the parameters and values are URL encoded.
- Multiple parameters are separated by an ampersand '&'.
- The following parameters can be included in the URL:
  - name
  - description
  - created
  - updated
  - process
  - category
  - operation_type
  - event
  - human_service
  - implementation_type
Retrieve an orchestration action entry
Use this REST API method to retrieve a particular orchestration action entry.

Available HTTP method

**Table 83. Retrieve an orchestration action entry REST API call**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/automation/%5Bid">https://hostname/resources/automation/[id</a>]</td>
</tr>
</tbody>
</table>

Response

JSON object for the specific orchestration action entry, based on the ID provided in the query:

```json
{  id:
   name:
   description:
   created:
   updated:
   process:
   process_app_id:
   process_app_short_name:
   process_app_name:
   category:
   operation_type:
   apply_to_all_pattern:
   event:
   icon:
   human_service:
   human_service_app_id:
   human_service_app_short_name:
   human_service_app_name:
   implementation_type:
   ownerid:
   pattern[
   {  
     id,
     patternId,
     patternType
   }
  ]
  priority:
}
```

Return values

- 200 - OK
- 401 - Unauthorized
- 404 - Not found

Add or update an entry in the orchestration actions
Use this method to update one of your entries in the orchestration actions or to add a new one.

Available HTTP method

**Table 84. Add or update an entry in the orchestration actions REST API call**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>PUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/automation/%5Bid">https://hostname/resources/automation/[id</a>]</td>
</tr>
</tbody>
</table>
### Table 84. Add or update an entry in the orchestration actions REST API call (continued)

<table>
<thead>
<tr>
<th>Response</th>
<th>JSON object for the updated orchestration action entry:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{ id:</td>
</tr>
<tr>
<td></td>
<td>name:</td>
</tr>
<tr>
<td></td>
<td>description:</td>
</tr>
<tr>
<td></td>
<td>process:</td>
</tr>
<tr>
<td></td>
<td>process_app_id:</td>
</tr>
<tr>
<td></td>
<td>process_app_short_name:</td>
</tr>
<tr>
<td></td>
<td>process_app_name:</td>
</tr>
<tr>
<td></td>
<td>category:</td>
</tr>
<tr>
<td></td>
<td>operation_type:</td>
</tr>
<tr>
<td></td>
<td>ownerid:</td>
</tr>
<tr>
<td></td>
<td>apply_to_all_pattern:</td>
</tr>
<tr>
<td></td>
<td>event:</td>
</tr>
<tr>
<td></td>
<td>human_service:</td>
</tr>
<tr>
<td></td>
<td>human_service_app_id:</td>
</tr>
<tr>
<td></td>
<td>human_service_app_short_name:</td>
</tr>
<tr>
<td></td>
<td>human_service_app_name:</td>
</tr>
<tr>
<td></td>
<td>implementation_type</td>
</tr>
<tr>
<td></td>
<td>pattern[</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>id:</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>paternType</td>
</tr>
<tr>
<td></td>
<td>]</td>
</tr>
</tbody>
</table>

| Return values | • 201 - Created and location (URL) of the updated orchestration action entry |
|               | • 401 - Unauthorized |
|               | • 404 - Not found |
|               | • 415 - Unsupported media type (incorrect XML document sent) |

---

**Delete an orchestration action entry**

Use this REST call to delete a selected entry from the orchestration actions.

### Available HTTP method

**Table 85. Delete an orchestration action entry REST API call**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/automation/%7Bid%7D">https://hostname/resources/automation/{id}</a></td>
</tr>
<tr>
<td>Response</td>
<td>The specified entry is deleted from the orchestration actions.</td>
</tr>
<tr>
<td>Return values</td>
<td>• 204 - No content</td>
</tr>
<tr>
<td></td>
<td>• 401 - Unauthorized</td>
</tr>
<tr>
<td></td>
<td>• 404 - Not found</td>
</tr>
</tbody>
</table>
Get entries for a specific virtual system

Use this REST call to retrieve information about orchestration action entries for a virtual system with a specified ID.

Table 86. Get entries for a specific virtual system REST API call

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/instances/%7Bid%7D/automation?parm1=value1&amp;parm2=value2">https://hostname/resources/instances/{id}/automation?parm1=value1&amp;parm2=value2</a>...</td>
</tr>
</tbody>
</table>

Response

JSON array of all entries from the orchestration actions, filtered according to the specified parameters:

```
{
  id:
  name:
  description:
  process:
  category:
  operation_type:
  apply_to_all_pattern:
  event:
  human_service:
  implementation_type
}
```

Return values

- 201 - Created and location (URL) of the updated automation registry entry
- 401 - Unauthorized
- 404 - Not found
- 415 - Unsupported media type (incorrect XML document sent)

Structure of the query string:

- A parameter list is appended to the URL after the '?' mark.
- The query parameters are specified as argument=value, where the arguments and values are URL encoded.
- Multiple parameters are separated by an ampersand &.
- The following parameters can be included in the URL:
  - name
  - description
  - process
  - category
  - operation_type
  - event
  - human_service
  - implementation_type
Business Process Manager Invoker REST API

You can use this set of Invoker REST APIs to retrieve information about artifacts that are available in Business Process Manager without accessing them directly.

For detailed information about Business Process Manager REST API, see the Business Process Manager information center.

Retrieve available BPM Business Processes

Use these APIs to retrieve a list of all the BPM Business Processes that are available for the implementation of self-service offerings or orchestration actions.

List all BPM Business Processes:

Use this REST call to retrieve a list of all the BPM Business Processes that are available for the implementation of self-service offerings or orchestration actions.

Available HTTP method

Table 87. Get list of all BPM Business Processes

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/bpm/runbook/">https://hostname/kernel/bpm/runbook/</a></td>
</tr>
</tbody>
</table>

Response

A list of available BPM Business Processes is returned. If no BPM Business Processes are available, an empty list is returned with HTTP Code 200. The returned list has the following parameters:

```json
{ id: displayName: processAppId: }
```

Return values

- 200 - No available BPM Business Processes found

An entry has the following attributes:

- **id** - the unique ID of the BPM Business Process as used in the underlying execution engine. Communication with the underlying engine, for example, to start a BPM Business Process, is typically done by using this ID.

- **displayName** - a human-readable name of the BPM Business Processes, typically used for display in the UI.

- **processAppId** - an identifier of a collection of BPM Business Processes to which the Business Process belongs.

The following listing shows an example response that can be retrieved via the request

GET /kernel/bpm/runbook/25.916d4552-9cf4-40c3-89fe-7f7bc43b2435:

```json
[
  {
    "id": "25.916d4552-9cf4-40c3-89fe-7f7bc43b2435",
    "displayName": "Create Business Object Test",
    "processAppId": "2066.5d35fbc5-6949-4971-8e06-83ca4c3cc760",
  },
  {
    "id": "25.2951bb80-e9b2-457a-9097-4443886d1dd5",
  }
]
```
Get entries for a specific BPM Business Process:

Use this REST call to retrieve information about a BPM Business Process with an indicated ID.

Available HTTP method

**Table 88. Get information about a specific BPM Business Process**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/bpm/runbook/runbook_id">https://hostname/kernel/bpm/runbook/runbook_id</a></td>
</tr>
</tbody>
</table>
| Response    | The following parameters of the BPM Business Process are retrieved:
  
  ```
  { id:
  displayName:
  processAppId:
  }
  ```
| Return values | 404 - The BPM Business Process not found |

This example shows the response to the following request GET /kernel/bpm/runbook/25.916d4552-9cf4-40c3-89fe-7f7bc43b2435:

```json
{
  "id": "25.916d4552-9cf4-40c3-89fe-7f7bc43b2435",
  "displayName": "Create Business Object Test",
  "processAppId": "2066.5d35fbc5-6949-4971-8e06-83ca4c3cc760",
}
```

Retrieve available human services

Use these APIs to retrieve information about the human services that are available for the implementation of self-service offerings and orchestration actions in IBM Cloud Orchestrator.

**List all human services:**

Use this REST API to retrieve a list of all the human services that are available for the implementation of self-service offerings and orchestration actions in IBM Cloud Orchestrator.

**Available HTTP method**

**Table 89. Get list of all human services**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/bpm/humanService">https://hostname/kernel/bpm/humanService</a></td>
</tr>
</tbody>
</table>
Table 89. Get list of all human services (continued)

<table>
<thead>
<tr>
<th>Response</th>
<th>A list of available human services is returned. If no human services are available, an empty list is returned with HTTP Code 200. The returned list has the following parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{ id: displayName: runUrl: }</td>
</tr>
<tr>
<td>Return values</td>
<td>• 200 - No available human services found</td>
</tr>
</tbody>
</table>

An entry has the following attributes:

- **id** - the unique ID of the human service as used in the underlying execution engine.
- **displayName** - a human-readable name of the human service, which is typically used for display in the UI.
- **runUrl** - the URL at which the human service can be started.

The following listing shows an example response that can be retrieved via the REST API:

```
[{
    "id": "1.6b0f42d8-0d65-4073-83bd-a3c7cb046f32",
    "displayName": "AddUserToVM",
    "runUrl": "http://xvm192:9080/teamworks/executeServiceByName?processApp=SCO_P1&serviceName=AddUserToVM"
},
{
    "id": "1.10027723-6b52-46f8-9105-535c75a09970",
    "displayName": "Show_Topology_Data",
    "runUrl": "http://xvm192:9080/teamworks/executeServiceByName?processApp=SCO_P1&serviceName=Show_Topology_Data"
}
]```

Get entries for a specific human service:

Use this REST call to retrieve information about a human service with an indicated ID.

**Available HTTP method**

Table 90. Get information about a specific human service

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/bpm/humanService/">https://hostname/kernel/bpm/humanService/</a> &lt;human_service_id&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>The following parameters of the human service are retrieved:</td>
</tr>
<tr>
<td></td>
<td>{ id: displayName: processAppId: }</td>
</tr>
<tr>
<td>Return values</td>
<td>• 404 - The human service not found</td>
</tr>
</tbody>
</table>
This example shows the response to the following request GET /kernel/bpm/
humanService /1.6b0f42d8-0d65-4073-83bd-a3c7cb046f32:
{
  "id": "1.6b0f42d8-0d65-4073-83bd-a3c7cb046f32",
  "displayName": "AddUserToVM",
  "runUrl": "http://xvm192:9080/teamworks/executeServiceByName?
    processApp=SCO_P1&serviceName=AddUserToVM" }

**Retrieve the Inbox**

Use these APIs to retrieve information about the contents of the Inbox.

For detailed information about Business Process Manager task REST API, see the
Business Process Manager information center.

**List all Inbox items:**

Use this REST API to retrieve a list of all the items contained in the Inbox.

**Available HTTP method**

*Table 91. Get list of all Inbox items*

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/bpm/task">https://hostname/kernel/bpm/task</a></td>
</tr>
</tbody>
</table>

**Response**

A list of Inbox items is returned. If Inbox is empty, an empty list is returned with HTTP Code 200. The returned list has the following parameters:

```c
{
  id:
  assignedTo:
  assignedToType:
  displayName:
  domain:
  operationContextId:
  project:
  relatedTo:
  requester:
  serviceInstanceId:
  taskDueDate
  taskOverdue:
  taskPriority:
  taskStatus:
  taskType:
  time:
}
```

**Return values**

- 200 - No available pending human activities found

An entry has the following attributes:

- **id** - the unique ID of the pending human activity as used in the underlying execution engine.
- **assignedTo** - the human-readable display name of the group or user to which this request is assigned.
- **assignedToType**:
  - **group** - if the task is still unclaimed.
  - **user** - if the task is claimed.
- **displayName** - a human-readable name of the pending human activity, which is typically used for display in the UI.
- **domain** - the domain of the user who requested this process.
- **operationContextId** - the ID of the operation context or request that is associated to this inbox.
- **project** - the project on behalf of which this process was requested.
- **relatedTo** - the name of the process to which the pending human activity belongs.
- **requester** - the requester of the process to which the pending activity belongs.
- **serviceInstanceId** - the ID of the associated virtual system instance, if the approval is part of a triggered event or user action.
- **taskDueDate** - due date of the pending human activity.
- **taskOverdue**:  
  - `true` - if the current date is later than the due date of the pending human activity.
  - `false` - otherwise.
- **taskPriority** - the priority of the task as used in the underlying execution engine.
- **taskStatus** - the status of the pending human activity as used in the underlying execution engine.
- **taskType** - the type of human activity:
  - `approval` - for an approval request.
  - `general` - for a general human task.
- **time** - the time at which the process was triggered.

The following listing shows an example response with one pending task:

```
[
  {
    "relatedTo": "Sample_DeleteInstanceApproval",
    "taskStatus": "Received",
    "taskPriority": "Normal",
    "taskOverdue": "true",
    "id": "8",
    "requester": "admin",
    "taskDueDate": "2013-08-19T17:27:26Z",
    "time": "2013-08-19T16:27:26Z",
    "displayName": "Delete Instance Approval: Sample1",
    "taskType": "approval",
    "assignedToType": "group",
    "operationContextId": "1007",
    "domain": "Default",
    "serviceInstanceId": null,
    "assignedTo": "All Users",
    "project": "admin"
  }
]
```
Get entries for a specific Inbox item:

Use this REST call to retrieve information about an Inbox item with an indicated ID

Available HTTP method

Table 92. Get information about a specific Inbox item

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/bpm/task/">https://hostname/kernel/bpm/task/</a>&lt;task id&gt;</td>
</tr>
</tbody>
</table>

Response

The following parameters of an Inbox item are retrieved:

```json
{
  overdue:
  dueDate:
  status:
  priority:
  id:
  displayName:
  requester:
  time:
  type:
  assignedToType:
  assignedTo:
  operationContextId:
  domain:
  project:
  serviceInstanceId:
  parameters:
    [
      {
        "Operation Type":
        "Operation Context ID":
        "Virtual System Pattern ID"
        "Virtual System Name"
        "Virtual System ID"
      },
    ],
  }
```

Return values

- 404 - The Inbox item not found

An entry has the following attributes:

- **overdue:**
  - true - if the current date is later than the due date of the pending human activity.
  - false - otherwise.
- **duedate** - due date of the pending human activity.
- **status** - the status of the pending human activity as used in the underlying execution engine.
- **priority** - the priority of the task as used in the underlying execution engine.
- **id** - the unique ID of the pending human activity as used in the underlying execution engine.
- **displayName** - a human-readable name of the pending human activity, which is typically used for display in the UI.
- **requester** - the requester of the process to which the pending activity belongs.
- **time** - the time at which the process was triggered.
- **type** - the type of human activity:
- approval - for an approval request
- general - for a general task

- **assignedToType:**
  - group - if the task is still unclaimed.
  - user - if the task is claimed.

- **assignedTo** - the human-readable display name of the group or user to which this request is assigned.

- **operationContextId** - the ID of the operation context or request that is associated to this inbox

- **domain** - the domain of the user who requested this process.

- **project** - the project on behalf of which this process was requested.

- **serviceInstanceId** - the ID of the associated virtual system instance, if the approval is part of a triggered event or user action.

- **parameters** - a name-value list that holds more information about the task.

  **Note:** For a general task, the parameters array is typically empty because an additional UI of the underlying process engine is provided which can be started via a URL.

This example shows the response to a request:

```json
{
  "overdue": "true",
  "dueDate": "2013-08-19T17:27:26Z",
  "status": "Pending",
  "priority": "Normal",
  "type": "approval",
  "id": "8",
  "requester": "admin",
  "time": "2013-08-19T16:27:26Z",
  "displayName": "Delete Instance Approval: Sample1"
  "assignedToType": "group",
  "operationContextId": "1007",
  "domain": "Default",
  "serviceInstanceId": null,
  "assignedTo": "All Users",
  "project": "admin",
  "parameters": [
    { "Operation Type": "Delete Instance" },
    { "Operation Context ID": "\kernel\tasks\505dce6d-7a97-4563-8f7b-6569ebc9e94c" },
    { "Virtual System Pattern ID": "\resources\patterns\1" },
    { "Virtual System Name": "Sample1" },
    { "Virtual System ID": "\resources\virtualSystems\2" }
  ]
}
```

### Service instance REST API

You can use this set of REST API calls to interact with the service instance resource.
Get entries for a specific service instance

Use this REST call to retrieve information about a specific service instance entry with a specified deployment ID.

Available HTTP method

Table 93. Get entries for a specific service instance entry with a specified deployment ID

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/serviceInstance/%7Bdeployment-id%7D">https://hostname/kernel/serviceInstance/{deployment-id}</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>The response has the following parameters:</td>
</tr>
<tr>
<td>`{</td>
</tr>
<tr>
<td>&quot;metaData&quot;: {</td>
</tr>
<tr>
<td>&quot;cloudGroup&quot;:</td>
</tr>
<tr>
<td>&quot;creator&quot;:</td>
</tr>
<tr>
<td>&quot;id&quot;:</td>
</tr>
<tr>
<td>&quot;name&quot;:</td>
</tr>
<tr>
<td>&quot;params&quot;:</td>
</tr>
<tr>
<td>&quot;status&quot;:</td>
</tr>
<tr>
<td>&quot;type&quot;:</td>
</tr>
<tr>
<td>&quot;virtualApplicationId&quot;:</td>
</tr>
<tr>
<td>&quot;virtualApplicationPatternId&quot;:</td>
</tr>
<tr>
<td>&quot;virtualSystemId&quot;:</td>
</tr>
<tr>
<td>&quot;virtualSystemPatternId&quot;: },</td>
</tr>
<tr>
<td>&quot;roles&quot;: [</td>
</tr>
<tr>
<td>&quot;services&quot;: [</td>
</tr>
<tr>
<td>&quot;virtualMachines&quot;: [</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>&quot;cloudGroup&quot;:</td>
</tr>
<tr>
<td>&quot;disk&quot;:</td>
</tr>
<tr>
<td>&quot;hostname&quot;:</td>
</tr>
<tr>
<td>&quot;hypervisorId&quot;:</td>
</tr>
<tr>
<td>&quot;id&quot;:</td>
</tr>
<tr>
<td>&quot;memory&quot;:</td>
</tr>
<tr>
<td>&quot;name&quot;:</td>
</tr>
<tr>
<td>&quot;networkInterfaces&quot;: [</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>&quot;hostname&quot;:</td>
</tr>
<tr>
<td>&quot;ip&quot;:</td>
</tr>
<tr>
<td>&quot;ipgroup&quot;:</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>]</td>
</tr>
<tr>
<td>]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Return values</th>
</tr>
</thead>
<tbody>
<tr>
<td>• 200 - OK</td>
</tr>
<tr>
<td>• 404 - The service instance with the given ID does not exist</td>
</tr>
<tr>
<td>• 500 - Internal server error</td>
</tr>
</tbody>
</table>

An entry has the following attributes:

- **metaData** - an object that contains instance level information, for example the name of the instance, its unique identifier, type, owner, and access information.
- **virtualMachines** - an array of objects that contains an object for each managed virtual machine.
• roles - an array that contains an object for each role. A role is a middleware role that a virtual machine plays among the different virtual machines in the deployed virtual application pattern.

• services - an array that contains an object for each service. A service is a managed entity other than a virtual machine or a role.

metaData has the following attributes:
• name - a string that is a human-readable name of the instance.
• type - a string that is internal (non-localized). It has one of the following values:
  – SINGLE_IMAGE_DEPLOYMENT - for instances that are deployed through the Virtual Images application.
  – TOPOLOGY - for instances that are deployed from a virtual system pattern.
  – APPLICATION - for instances that are deployed from a virtual application pattern or shared service.
• status - a string that is internal (non-localized). In case of the TOPOLOGY or SINGLE_IMAGE_DEPLOYMENT type, the same as for the virtual system state machine. In case of other types, the same as for the virtual application state machine.
• id - the URI to retrieve the most current version of the document.
• virtualSystemId - the URI to retrieve the virtual system if there is any that is associated.
• virtualApplicationId - the URI to retrieve the virtual application instance if there is any that is associated.
• virtualApplicationPatternId - the storehouse URI for the vApp pattern if there is any that is associated.
• virtualSystemPatternId - the URI for the system pattern that is used to create if there is any that is associated.
• creator - the ID of the user who initiated the deployment of the instance. This id is a resource id like /resources/users/1 for vSys and single image deployments, and a storehouse id like /storehouse/admin/users/u-0 in other cases.
• cloudGroup - the cloud group to which the virtual application instance was deployed. The field can be empty for virtual system pattern deployments, or single image deployments.
• params - a JSON object that contains custom parameters that were provided by extension writers.

virtualMachine has the following attributes:
• name - a string identifier of the virtual machine. Typically, it does not match the primary host name of the virtual machine. The identifier is unique within a service instance.
• id - a unique string identifier of the virtual machine that is relative to the service instance base url.
• cloudGroup - the URI of the cloud group where the virtual machine is deployed.
• networkInterfaces - network interfaces of the virtual machine.
• hostname - the primary host name of the virtual machine. If there are multiple network interfaces, the primary host name depends on the implementation.
• virtualCpu - the number of virtual central processing units that is the number of central processing units that the guest operating system on this virtual machine sees.
- **memory** - the amount of memory that the guest operating system sees. The unit of measurement is mebibyte (1 MB = 1048576 bytes).

- **disk** - the size of the primary/root disk. The unit of measurement is mebibyte (1 MB = 1048576 bytes).

- **imageId** - the image that was used to create the virtual machine. The lifecycles of the virtual machine and image are separate.

- **partname** - the name of the part in the associated virtual system pattern, from which the virtual machine was instantiated.

- **runtimeId** - the ID of the virtual machine on the hypervisor. The format of this string depends on the hypervisor.

- **params** - the custom parameters that were provided by extension writers.

**networkInterface** has the following attributes:

- **ip** - the IPv4 address in dotted decimal notation or the IPv6 address.

- **hostname** - a host name that should resolve to the given IP through DNS.

- **ipgroup** - the URI of the IP group that the address was allocated from.

**role** has the following attributes:

- **name** - the name of the role.

- **type** - the type of the role.

- **endpoints** - an array that contains an object for each endpoint.

- **params** - a JSON object that contains custom parameters that were provided by extension writers.

**service** has the following attributes:

- **name** - a human-readable string that identifies the service.

- **type** - either a java-style package identifier or a tosca node type.

- **params** - contain custom parameters that were provided by extension writers. See the details below:

  It is a free-form JSON object that holds additional parameters that are required by extension developers. The JSON object has the following limitations:

  - There is no support for JSON arrays, which are converted to strings.

  - All non-string simple types are converted to string when they are stored.

  You can read and edit these objects to add data that their extension requires in the context of a service instance. You can retrieve the data later from a Business Process Manager process or other extension. This storage is backed by the storehouse metadata APIs.

**endpoint** has the following attributes:

- **name** - a human-readable string that identifies the endpoint.

- **URI** - The URI that the endpoint points to.

The following listing shows an example response that can be retrieved by the request:

```json
{
  "metaData": {
    "cloudGroup": "\resource\{\clouds\}/1",
    "creator": "\resource\{\user\}/2",
    "id": "\kernel\serviceInstance\/1",
    "name": "Test",
    "params": {
```
"Hello": "World",
"status": "RM01005",
"type": "TOPOLOGY",
"virtualApplicationId": "",
"virtualApplicationPatternId": "",
"virtualSystemId": "\resources\virtualSystems\1",
"virtualSystemPatternId": "\resources\patterns\1"
},
"roles": [
],
"services": [
],
"virtualMachines": [
{
"cloudGroup": "\resources\clouds\1",
"disk": 10240,
"hostname": "172-0-0-13.lightspeed.brhmal.sbcglobal.net",
"hypervisorId": "\resources\hypervisors\PM-1",
"id": "1",
"memory": 2048,
"name": "965a92be-OS Node-Test-1",
"networkInterfaces": [
{
"hostname": "172-0-0-13.lightspeed.brhmal.sbcglobal.net",
"ip": "172.0.0.13",
"ipgroup": "\resources\ipgroups\2"
}
],
"partname": "OS Node",
"runtimeId": "1b9417ca-6f66-42ec-a2da-661739d8e6ed",
"virtualCpu": 1
}
}

## Metadata parameters REST API

You can use this set of REST API calls to interact with metadata parameters.

### Get parameters of specific metadata:

Use this REST call to retrieve information about metadata parameters of a specific service instance.

**Available HTTP method**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/serviceInstance/%7Bdeployment-id%7D/metaData/params/attribute.member.member">https://hostname/kernel/serviceInstance/{deployment-id}/metaData/params/attribute.member.member</a></td>
</tr>
<tr>
<td>Response</td>
<td>Returns the specified member of the object which is an attribute within the parameters. Only the specified member of the object is returned.</td>
</tr>
<tr>
<td>Return values</td>
<td>• 200 - OK&lt;br&gt;• 500 - Internal Server Error</td>
</tr>
</tbody>
</table>

**Tip:** To navigate within JSON objects, use . (dot).

The following listing shows an example response that can be retrieved via the request:
Post metadata parameters:

Use this REST API call to post metadata parameters of a specific service instance.

Available HTTP method

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>POST</th>
</tr>
</thead>
</table>

| URL pattern      | https://hostname/kernel/serviceInstance/{deployment-id}/metaData/params |
| Response         | Updates parameters partially. Merges the posted attributes into the existing parameters. Attributes that already exist are replaced with the new values. New attributes are created. |

| Return values    | • 200 - OK  
|                 | • 500 - Internal Server Error |

**Note:** For this rest call, you need a content-type: application/json header.

The following listing shows a sample request body:

```json
{"Hello":"World"}
```

The following listing shows the response:

```json
{"Status":"Ok"}
```

Delete parameters of specific metadata:

Use this REST API call to delete information about metadata parameters of a specific service instance.

Available HTTP method

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>DELETE</th>
</tr>
</thead>
</table>

| URL pattern                        | https://hostname/kernel/serviceInstance/{deployment-id}/metaData/params/attribute.member.member |
| Response                            | Deletes the specified member of the object that is an attribute within the parameters. Only the specified member of the object is deleted.  
|                                    |{"Status":"Ok"} |
| Return values                       | • 200 - OK  
|                                    | • 500 - No query parameter is specified, or internal server error |

**Note:** The REST call always returns a `{"Status":"Ok"}` response on a 200 return value.
Virtual machines parameters REST API

You can use this set of REST API calls to interact with virtual machines parameters.

Get parameters of a specific virtual machine:

Use this REST API call to retrieve information about virtual machine parameters of a specific service instance.

Available HTTP method

<table>
<thead>
<tr>
<th>Table 97. Get virtual machine parameters of a specific service instance REST API call</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP method</td>
</tr>
<tr>
<td>URL pattern</td>
</tr>
<tr>
<td>Response</td>
</tr>
</tbody>
</table>
| Return values | • 200 - OK  
• 500 - Internal Server Error |

Tip: To navigate within JSON objects, use . (dot).

The following listing shows an example response that can be retrieved via the request:

{"Hello":"World"}

Post virtual machine parameters:

Use this REST API call to post virtual machine parameters of a specific service instance.

Available HTTP method

<table>
<thead>
<tr>
<th>Table 98. Post virtual machine parameters of a specific service instance REST API call</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP method</td>
</tr>
<tr>
<td>URL pattern</td>
</tr>
<tr>
<td>Response</td>
</tr>
</tbody>
</table>
| Return values | • 200 - OK  
• 500 - Internal Server Error |

Note: For this rest call, you need a content-type: application/json header.

The following listing shows a sample request body:

{"Hello":"World"}

The following listing shows the response:

{"Status":"Ok"}
Delete parameters of specific virtual machines:

Use this REST API call to delete information about virtual machines parameters of a specific service instance.

Available HTTP method

**Table 99. Delete virtual machines parameters of a specific service instance REST API call**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/kernel/serviceInstance/%7BdeploymentID%7D/virtualMachines/%7Bname%7D/params/attribute.member.member">https://hostname/kernel/serviceInstance/{deploymentID}/virtualMachines/{name}/params/attribute.member.member</a></td>
</tr>
<tr>
<td>Response</td>
<td>Deletes the specified member of the object that is an attribute within the parameters. Only the specified member of the object is deleted.</td>
</tr>
<tr>
<td>Return values</td>
<td>200 - OK</td>
</tr>
</tbody>
</table>

**Note:** The REST call always returns a \{"Status":"ok"\} response on a 200 return value.

Deployment parameters REST API

You can use this set of REST API calls to interact with deployment parameters of a specific instance.

Get deployment parameters for a specific instance

Use this REST API call to retrieve information about deployment parameters for a specific service instance.

Available HTTP method

**Table 100. Get deployment parameters for a specific service instance REST API call**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="http://hostname:port/resources/instances/%7Binstanceid%7D/deploymentparameters">http://hostname:port/resources/instances/{instanceid}/deploymentparameters</a></td>
</tr>
<tr>
<td>Response</td>
<td>JSON object with the deployment parameters for the virtual system instance filtered according to the query parameters:</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>parameterclass: parametername: parametervalue: partkey: userconfigurable: scriptpackagename: scriptpackagetype:</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>Return values</td>
<td>200 - OK</td>
</tr>
<tr>
<td></td>
<td>400 - Not Found</td>
</tr>
<tr>
<td></td>
<td>500 - Unexpected Error</td>
</tr>
</tbody>
</table>

**Note:** Instanceid is a virtual system ID (integer value).
Deployment parameters have following attributes:

- **parameterclass** - string, parameter class as defined by the underlying pattern model.
- **partkey** - string, label of the related virtual system part for the parameter.
- **parametername** - string, the name of the parameter.
- **parametervalue** - string, the value of the parameter.
- **userconfigurable** - boolean, specifies whether parametervalue can be updated.

Note: The combination of **parameterclass**, **partkey** and **parametername** is unique within a virtual system instance.

Deployment parameters of script packages have two additional attributes:

- **scriptpackagename** - string, the name of a script package.
- **scriptpackagetype** - string, the type of a script package

The attribute **scriptpackagetype** has two values:

- **APPLICATION** for custom script packages.
- **ADDON_<type>** for add-ons.

### Post deployment parameters for a specific instance

Use this REST API call to post information about deployment parameters for a specific service instance.

#### Available HTTP method

**Table 101. Post deployment parameters for a specific service instance REST API call**

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="http://hostname:port/resources/instances/%7Binstanceid%7D/deploymentparameters">http://hostname:port/resources/instances/{instanceid}/deploymentparameters</a></td>
</tr>
<tr>
<td>Response</td>
<td>JSON object with the deployment parameters for the virtual system instance that is filtered according to the query parameters:</td>
</tr>
<tr>
<td></td>
<td>{</td>
</tr>
<tr>
<td></td>
<td>parametername:</td>
</tr>
<tr>
<td></td>
<td>parametervalue:</td>
</tr>
<tr>
<td></td>
<td>partkey:</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>Return values</td>
<td>• 200 - OK</td>
</tr>
<tr>
<td></td>
<td>• 400 - Not Found</td>
</tr>
<tr>
<td></td>
<td>• 500 - Unexpected Error</td>
</tr>
</tbody>
</table>

Note: {instanceid} is a virtual system ID (integer value).

For the method to be successful, the following conditions must be met:

- The deployment parameter is known in the pattern.
- The deployment parameter is user-configurable.
- The call must be made before provisioning.

Deployment parameters have following attributes:

- **partkey** - string, label of the related virtual system part for the parameter.
- **parametername** - string, the name of the parameter.
- **parametervalue** - string, the value of the parameter.
Core Services REST API

Core Services REST API overview
IBM Cloud Orchestrator uses REST APIs to allow easy, lightweight communication between components and integration with external systems.

Linked Resources vs. Collection Resources

REST API implements a linked data concept where relations between resources are provided with the response as meta-data. A linked resource is the most basic entity in a response. It represents a resource itself as well as any links to other resources. The following code is a structure of a linked resource:

```json
{
  "href": "https://host:9443/orchestrator/v2/...",
  "item": { ... },
  "link_1": {
    "href": "https://host:9443/orchestrator/v2/..."
  },
  "link_2": {
    "href": "https://host:9443/orchestrator/v2/..."
  }
}
```

Each linked resource has at least one link to itself, the first href property. An item property follows with the actual resource representation. There can also be additional links to other resources. A collection resource is a collection of linked resources. In addition to the basic properties of a linked resource, a collection resource also features specific properties for pagination. The following code displays the structure of a collection resource:

```json
{
  "href": "https://host:9443/orchestrator/v2/collection",
  "start": 10,
  "limit": 10,
  "total": 49,
  "first": {
    "href": "https://host:9443/orchestrator/v2/collection/?_limit=10&_start=0"
  },
  "previous": {
    "href": "https://host:9443/orchestrator/v2/collection/?_limit=10&_start=0"
  },
  "next": {
    "href": "https://host:9443/orchestrator/v2/collection/?_limit=10&_start=20"
  },
  "last": {
    "href": "https://host:9443/orchestrator/v2/collection/?_limit=10&_start=39"
  },
  "items": [ ... ]
}
```

The start, limit and total properties enable you to display the correct number of pages in a UI. The number of pages is the total size divided by the page size. You can also choose to leverage the provided first, next, etc. links and can call them directly from a UI to navigate the collection easily.
HTTP Status Codes

The following codes apply:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Request</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 OK</td>
<td>GET /resource, GET /collection, PUT /resource</td>
<td>General status if the request went OK, no resources were created.</td>
</tr>
<tr>
<td>201 Created</td>
<td>POST /collection</td>
<td>New resource was created.</td>
</tr>
<tr>
<td>202 Accepted</td>
<td>POST /collection/{id}/launch</td>
<td>A launch request was accepted.</td>
</tr>
<tr>
<td>204 No Content</td>
<td>DELETE /resource</td>
<td>A resource was deleted.</td>
</tr>
<tr>
<td>400 Bad Request</td>
<td>PUT /resource, POST /collection</td>
<td>Request payload was not complete or badly formatted.</td>
</tr>
<tr>
<td>401 Unauthorized</td>
<td>Any</td>
<td>Session has expired or no SimpleToken was provided.</td>
</tr>
<tr>
<td>403 Forbidden</td>
<td>Any</td>
<td>Session is valid but user was not authorized for the requested operation.</td>
</tr>
<tr>
<td>404 Not Found</td>
<td>Any</td>
<td>Requested resource path was not found.</td>
</tr>
<tr>
<td>405 Method Not Allowed</td>
<td>PUT /collection, DELETE /collection</td>
<td>Tried to update or delete a collection resource.</td>
</tr>
<tr>
<td>406 Not Acceptable</td>
<td>Any</td>
<td>Invalid Accept header. Only application/json is allowed by default.</td>
</tr>
<tr>
<td>409 Conflict</td>
<td>POST /collection</td>
<td>A resource with the same identifier already exists.</td>
</tr>
<tr>
<td>415 Unsupported Media Type</td>
<td>POST /collection, PUT /resource</td>
<td>Requested invalid content type. Only application/json is allowed by default.</td>
</tr>
<tr>
<td>500 Internal Server Error</td>
<td>Any</td>
<td>An internal error occurred. Log files should be checked. The REST API may provide additional hints in the response body.</td>
</tr>
</tbody>
</table>

HTTP Media Types

By default, all REST APIs consume and produce application/json as their media type. Other types such as text/xml are not supported unless stated otherwise.

Pagination, Filtering, Sorting and Searching

URL parameters used to control the output of a REST API are generally prefixed with an underscore _ to distinguish them from queries on resource properties. This avoids confusion between

/apples?sort=goldendelicious (gives all apples of the "goldendelicious" sort)

and
The following table describes URL keywords controlling REST API behavior:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>_start=n</td>
<td>Start pagination at element n.</td>
</tr>
<tr>
<td>_limit=n</td>
<td>Set page size to n elements per page. There is an upper boundary of 100 items and a lower boundary of 5 items. If omitted, 10 is the default.</td>
</tr>
<tr>
<td>_sortby=abc</td>
<td>Sort result set by resource attribute abc. If omitted, id (or the respective identifier of the resource) is the default.</td>
</tr>
<tr>
<td>_sort=asc</td>
<td>desc</td>
</tr>
<tr>
<td>_search=abc</td>
<td>Do a case-insensitive, full-text search for abc in the searchable parts of a resource. Depends on REST API implementation, usually name and description are searchable.</td>
</tr>
<tr>
<td>property=value</td>
<td>If no underscore prefix was given, filter for resources with properties containing value. Multiple values can be passed as a separate property=value pair.</td>
</tr>
</tbody>
</table>

Examples

- `/orchestrator/v2/categories` - Returns service catalog categories starting at index 0 with a limit of 10, sorted by ID in ascending order.
- `/orchestrator/v2/categories?_start=10` - Returns service catalog categories starting at index 10 with a limit of 10, sorted by ID in ascending order.
- `/orchestrator/v2/categories?_start=10&_limit=20` - Returns service catalog categories starting at index 10 with a limit of 20, sorted by ID in ascending order.
- `/orchestrator/v2/categories?_sortby=name` - Returns service catalog categories starting at index 0 with a limit of 10, sorted by name in ascending order.
- `/orchestrator/v2/categories?_sortby=id&_search=virtual` - Returns service catalog categories containing the word "virtual" starting at index 0 with a limit of 10, sorted by ID in ascending order.
- `/orchestrator/v2/categories?name=OpenStack` - Returns service catalog categories whose name is "OpenStack" starting at index 0 with a limit of 10.
- `/orchestrator/v2/categories?id=123&id=456` - Returns service catalog categories with the ids 123 and 456 starting at index 0 with a limit of 10.
Offering REST API V2

The following topics cover categories, offering attributes and offering instances of the offering REST API V2.

Categories:

Json Formats

Category Request:

{
   "isbuiltin": 0,
   "icon": "Web Machine Category Icon:ge100_webcatalog_24",
   "name": "Manage Virtual Machines",
   "description": "Deploy, start, stop and virtual machines based on a single image."
}

Category Response:

{
   "href": "https://host:9443/orchestrator/v2/categories/4711",
   "item": {
      "id": 4711,
      "isbuiltin": 0,
      "icon": "Web Machine Category Icon:ge100_webcatalog_24",
      "name": "Manage Virtual Machines",
      "description": "Deploy, start, stop and virtual machines based on a single image."
   }
}

Categories Response

{
   "href": "https://host:9443/orchestrator/v2/categories/",
   "start": 0,
   "limit": 10,
   "total": 9,
   "first": {
      "href": "https://host:9443/orchestrator/v2/categories/?_limit=10&_start=0"
   },
   "previous": null,
   "next": null,
   "last": {
      "href": "https://host:9443/orchestrator/v2/categories/?_limit=10&_start=0"
   },
   "items": [ Category Response, ...., Category Response ]
}

Table 104.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Type</th>
<th>Mandatory</th>
<th>Generated</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>category id</td>
<td>Number</td>
<td>no</td>
<td>yes</td>
<td>automatically assigned when a new category gets created</td>
</tr>
</tbody>
</table>
Table 104. (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Type</th>
<th>Mandatory</th>
<th>Generated</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>icon</td>
<td>icon name</td>
<td>String</td>
<td>no</td>
<td>no</td>
<td>name of the icon that is displayed with the category in the UI</td>
</tr>
<tr>
<td>name</td>
<td>category name</td>
<td>String</td>
<td>yes</td>
<td>no</td>
<td>name of the category</td>
</tr>
<tr>
<td>description</td>
<td>category description</td>
<td>String</td>
<td>yes</td>
<td>no</td>
<td>description of the category</td>
</tr>
<tr>
<td>isbuiltin</td>
<td>built in</td>
<td>Number</td>
<td>no</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GET: Lists offering categories

**URL pattern**
/orchestrator/v2/categories

**Accepts**
*  

**Content-Type**
application/JSON

**Normal Response Codes**
200 OK

**Error Response Codes**
401 unauthorized
500 internal server error

**Response**
Categories Response

**Search Attributes**
name, description

**Filter Attributes**
all

**Authorization**
no authorization needed

POST: Creates a offering category

**URL pattern**
/orchestrator/v2/categories

**Accepts**
application/JSON

**Content-Type**
application/JSON

**Normal Response Codes**
201 created
Error Response Codes
400 bad request if bad JSON was passed or mandatory attributes were missing
401 unauthorized
500 internal server error

Request
Category Request

Response
Category Response

Authorization
role:"admin"

GET: Get category

URL pattern
/orchestrator/v2/categories/{id}

Accepts
*

Content-Type
application/JSON

Normal Response Codes
200 OK

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Response
Category Response

Authorization
no authorization

PUT: Update category

URL pattern
/orchestrator/v2/categories/{id}

Accepts
application/JSON

Content-Type
application/JSON

Normal Response Codes
200 OK

Error Response Codes
400 bad request if bad JSON was passed
401 unauthorized
404 not found
500 internal server error
Request
{
...
"name": "Manage Virtual Image",
"description": "Deploy, start, stop"
...
}

Response
Category Response

Authorization
role: "admin"

DELETE: Delete category

URL pattern
/orchestrator/v2/categories/{id}

Accepts
*

Normal Response Codes
204 no content

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Authorization
role: "admin"

Offering attributes:

Attributes for the offering are displayed in this section.

Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Type</th>
<th>Mandatory</th>
<th>Generated</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>service id</td>
<td>Number</td>
<td>no</td>
<td>yes</td>
<td>automatically assigned when a new offering gets created</td>
</tr>
<tr>
<td>icon</td>
<td>icon name</td>
<td>String</td>
<td>no</td>
<td>no</td>
<td>name of the icon that is displayed with the category in the UI</td>
</tr>
<tr>
<td>name</td>
<td>offering name</td>
<td>String</td>
<td>yes</td>
<td>no</td>
<td>name of the offering</td>
</tr>
<tr>
<td>description</td>
<td>offering description</td>
<td>String</td>
<td>yes</td>
<td>no</td>
<td>description of the offering</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
<td>Type</td>
<td>Mandatory</td>
<td>Generated</td>
<td>Comment</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>category</td>
<td>offering</td>
<td>Number</td>
<td>no</td>
<td>no</td>
<td>category id of this offering</td>
</tr>
<tr>
<td>implementation_type</td>
<td>process that gets started</td>
<td>String</td>
<td>no</td>
<td></td>
<td>defaults to &quot;ibm_bpm_process&quot; if not passed</td>
</tr>
<tr>
<td>process_app_id</td>
<td>BPM process app id containing the linked process (process attribute)</td>
<td>String</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>process</td>
<td>BPM process implementing the offering or action</td>
<td>String</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>human_service_app_id</td>
<td>BPM process app id containing the linked human service (human_service attribute)</td>
<td>String</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>human_service</td>
<td>Human service implementing the User Interface for the offering or action</td>
<td>String</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ownerid</td>
<td></td>
<td>no</td>
<td></td>
<td></td>
<td>????</td>
</tr>
<tr>
<td>operation_type</td>
<td></td>
<td>String</td>
<td>no</td>
<td></td>
<td>&quot;offering&quot;, &quot;singleInstanceAction&quot;, &quot;multiInstanceAction&quot;</td>
</tr>
<tr>
<td>instancetype</td>
<td>type of instances the process is working on</td>
<td>String</td>
<td>no</td>
<td>no</td>
<td>name of instance provider</td>
</tr>
<tr>
<td>tags</td>
<td>List of service designer tags matching to the ones of the instance type</td>
<td>List of Strings</td>
<td>no</td>
<td>no</td>
<td>subset of tags of instance provider</td>
</tr>
<tr>
<td>acl</td>
<td>Access control list</td>
<td>List of ACL JSON</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>acl/domain</td>
<td></td>
<td></td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acl/project</td>
<td></td>
<td></td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acl/role</td>
<td></td>
<td></td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acl/use</td>
<td></td>
<td></td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>acl/modify</td>
<td></td>
<td></td>
<td>no</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Offering instances:

A list of instances for the offering are described in this section.

Instances

GET: Lists offerings

URL pattern
/orchestrator/v2/offerings

Accepts
*

Content-Type
application/JSON

Normal Response Codes
200 OK

Error Response Codes
401 unauthorized
500 internal server error

Response
Offerings Response

Search Attributes
name, description

Filter Attributes
id, name, description, icon, human_service, human_service_app_id,
priority, created, updated, process, process_app_id, owner_id, category,
implementation_type, operation_type, instancetype

Authorization
role: admin or ACL with 'view' set to 'true' for given domain, project and
role

POST: Creates an offering

URL pattern
/orchestrator/v2/offerings

Accepts
application/JSON

Content-Type
application/JSON

Normal Response Codes
201 created

Error Response Codes
400 bad request if bad JSON was passed or mandatory attributes were
missing
401 unauthorized
500 internal server error

Request
Offering Request

Response
Offering Response

Authorization
roles: "admin", "domain_admin"

POST: Creates an offering

URL pattern
/orchestrator/v2/offерings

Accepts
application/JSON

Content-Type
application/JSON

Normal Response Codes
201 created

Error Response Codes
400 bad request if bad JSON was passed or mandatory attributes were missing
401 unauthorized
500 internal server error

Request
Offering Request

Response
Offering Response

Authorization
roles: "admin", "domain_admin"

GET: Get an offering

URL pattern
/orchestrator/v2/offерings/{id}

Accepts
*

Content-Type
application/JSON

Normal Response Codes
200 OK

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Response
Offering Response
Authorization
role: admin or ACL with 'view' set to 'true' for given domain, project and role

PUT: Update an offering
URL pattern
/orchestrator/v2/offerings/{id}
Accepts
application/JSON
Content-Type
application/JSON
Normal Response Codes
200 OK
Error Response Codes
400 bad request if bad JSON was passed
401 unauthorized
404 not found
500 internal server error

Request
Offering Request (partial)

Response
Offering Response

Authorization
role: admin or ACL with 'modify' set to 'true' for given domain, project and role

DELETE: Delete an offering
URL pattern
/orchestrator/v2/offerings/{id}
Accepts *

Normal Response Codes
204 no content

Error Response Codes
401 unauthorized
500 internal server error

Authorization
role: admin or ACL with 'modify' set to 'true' for given domain, project and role

POST: Executes and offering
URL pattern
/orchestrator/v2/offerings/{id}/launch
Accepts
application/JSON
Content-Type
application/JSON

Normal Response Codes
202 accepted

Error Response Codes
400 bad request if bad JSON was passed
401 unauthorized
404 not found
500 internal server error

Request
LAunchRequest TBD

Response
TaskResponse

Authorization
role: admin or ACL with 'use' set to 'true' for given domain, project and role

GET: Get ACL entries for a given offering

URL pattern
/orchestrator/v2/offering/{id}/acl

Accepts
*

Content-Type
application/JSON

Normal Response Codes
200 OK

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Response
ACLs Response

Authorization
no authorization needed but result is restricted for the given domain, project and role of the user

PUT: Update given acl for a given offering

URL pattern
/orchestrator/v2/offering/{id}/acl

Accepts
application/JSON

Content-Type
application/JSON

Normal Response Codes
200
Error Response Codes
400 bad request of bas JSON was passed
401 unauthorized
404 not found
500 internal server error

Request
ACLS Request

Response
ACLS Response

Authorization
no authorization needed but the given ACL is adjusted to the given
domain, project and role of the user

GET: Get input parameters for a given offering

URL pattern
/orchestrator/v2/offers/[id]/parameters

Accepts
*

Content-Type
application/JSON

Normal Response Codes
200 OK

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Response
Parameters Response

Authorization
no authorization needed but result is restricted for the given domain,
project and role of the user

GET: Get graphical representation of a given offering workflow

URL pattern
/orchestrator/v2/offers/[id]/graph

Accepts
*

Content-Type
application/JSON

Normal Response Codes
200 OK

Error Response Codes
401 unauthorized
404 not found
500 internal server error
Response
image/jpeg binary image representing this offering's workflow

Authorization
no authorization needed but result is restricted for the given domain, project and role of the user

Resource Instances REST API
Instances returned for a specific type are collected using an instance provider. An instance provider is a Java class that can talk to a back-end to query instance information such as VMs, disks, users, networks and other cloud resources.

JSON Formats

Resource Type Request

```
{
  "name" : "myprovider",
  "displayname" : "My Provider",
  "description" : "This is my provider",
  "icon" : "Web Icon:glyphicons_266_flag",
  "provider" : "com.ibm.orchestrator.core.instance.providers.myprovider.MyProvider",
  "type" : "admin",
  "tags" : ["enabled", "disabled"],
  "detailsview" : {
    "application" : "SCOABC",
    "humanservice" : "Show My Provider Details"
  },
  "keyfields" : [{
    "instanceattribute" : "displayname",
    "header" : "Name"
  }, {
    "instanceattribute" : "description",
    "header" : "Description"
  }
}
```

Resource Type Response

```
{
  "name" : "myprovider",
  "displayname" : "My Provider",
  "description" : "This is my provider",
  "icon" : "Web Icon:glyphicons_266_flag",
  "provider" : "com.ibm.orchestrator.core.instance.providers.myprovider.MyProvider",
  "type" : "admin",
  "tags" : ["enabled", "disabled"],
  "detailsview" : {
    "application" : "SCOABC",
    "humanservice" : "Show My Provider Details"
  },
  "keyfields" : [{
    "instanceattribute" : "displayname",
    "header" : "Name"
  }, {
    "instanceattribute" : "description",
    "header" : "Description"
  }
}
```

Resource Types Response
Resource Instance Request
{
  "parm":
  {
    .... Instance type dependent parameter JSON object ....
  },
  "displayname": "mhtest1",
  "detailsURL": "<hostname:port>/teamworks/executeServiceByName?processApp=<FOO> &serviceName=Show+Server+Details&tw.local.serverId=b479108c-df8f-4462-be0b-f80af4a59d15&tw.local.region=RegionOne&tw.local.user=<user>&tw.local.domain=Default&tw.local.project=<project>",
  "status": "ACTIVE",
  "region": "RegionOne",
  "icon": "Server Category Icon:ge100_servercatalog_24",
  "openstackId": "b479108c-df8f-4462-be0b-f80af4a59d15",
  "tags":
  [ active 
  ],
  "id": "RegionOne--b479108c-df8f-4462-be0b-f80af4a59d15",
  "updated": "2014-03-31T11:04:58Z",
  "ipAddresses": "vmnet: 10.0.0.100",
  "description": "mhtest1"
}

Resource Instance Response
{
  "href": "<hostname:port>/orchestrator/v2/instancetypes/openstackvms/instances/RegionOne--b479108c-df8f-4462-be0b-f80af4a59d15",
  "created": "2014-03-31T11:04:18Z",
  "parm":
  {
    .... Instance type dependent parameter JSON object ....
  },
  "displayname": "mhtest1",
  "detailsURL": "<hostname:port>/teamworks/executeServiceByName?processApp=<FOO> &serviceName=Show+Server+Details&tw.local.serverId=b479108c-df8f-4462-be0b-f80af4a59d15&tw.local.region=RegionOne&tw.local.user=<user>&tw.local.domain=Default&tw.local.project=<project>",
  "status": "ACTIVE",
  "region": "RegionOne",
  "icon": "Server Category Icon:ge100_servercatalog_24",
  "openstackId": "b479108c-df8f-4462-be0b-f80af4a59d15",
  "tags":
  [ active 
  ],
  "id": "RegionOne--b479108c-df8f-4462-be0b-f80af4a59d15"
Resource Instances Response
{
    "href": "<hostname:port>/orchestrator/v2/instancetypes/openstackvms/instances",
    "start": 0,
    "limit": 10,
    "total": 2,
    "first": "<hostname:port>/orchestrator/v2/instancetypes/openstackvms/instances?_start=0&_limit=10",
    "previous": null,
    "next": null,
    "last": "<hostname:port>/orchestrator/v2/instancetypes/openstackvms/instances?_start=0&_limit=10",
    "items":
        [
            Resource Instance Response 1, ..., Resource Instance Response n
        ]
}

Instances

GET : Lists all resource types

URL pattern
/orchestrator/v2/instancetypes

Accepts
*/*

Content-Type
application/JSON

Normal Response Codes
200

Error Response Codes
500 internal server error

Response
Resource Type Response

Authorization
No authorization needed

POST: Create resource type

URL pattern
/orchestrator/v2/instancetypes/

Accepts
application/JSON

Content type
application/JSON

Normal Response Codes
201

Error Response Codes
401 unauthorized
409 conflict
500 internal server error

Request
Resource Type Request

Response
Resource Type Request

Authorization
role: admin

GET: Get one resource type

URL pattern
/orchestrator/v2/instancetypes/{name}

Accepts
*/*

Content-Type
application/JSON

Normal Response Codes
200

Error Response Codes
404 not found
500 internal server error

Response
Resource Type Response

Authorization
No authorization needed

PUT: Update a resource type

URL pattern
/orchestrator/v2/instancetypes/{name}

Accepts
application/JSON

Content-Type
application/JSON

Normal Response Codes
200

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Request
Resource Type Request (partial)

Response
Resource Type Response

Authorization
role: admin
DELETE: Delete a resource type.

URL pattern
/orchestrator/v2/instancetypes/{name}

accepts
/*/*

Content-Type
application/JSON

Normal Response Codes
204

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Authorization
role: admin

GET: List instances of a given type.

URL pattern
/orchestrator/v2/instancetypes/{name}/instances

Accepts
/*/*

Content-Type
application/JSON

Normal Response Codes
200

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Response
Resource Instances Response

Authorization
Instance provider dependent. Generic Provider: Access Control Link with view set to true for the domain, project and role you are working on.

POST: Creates a instance of a given type.

URL pattern
/orchestrator/v2/instancetypes/{name}/instances

Accepts
application/JSON

Content-Type
application/JSON

Normal Response Codes
201 created
Error Response Codes
   401 unauthorized
   404 not found
   500 internal server error

Request
   Resource Instance Request

Response
   Resource Instance Response

Authorization
   Instance provider dependent. Generic Provider: role: ADMIN

GET: Gets an instance of a given type.

URL pattern
   /orchestrator/v2/instancetypes/{name}/instances/{id}

Accepts
   application/JSON

Content-Type
   */*

Normal Response Codes
   401 unauthorized
   404 not found
   500 internal server error

Response
   Resource Instance Response

Authorization
   Instance provider dependent. Generic Provider: Access Control Link with
   view set to true for the domain, project and role you are working on.

PUT: Updates an instance of a given type.

URL pattern
   /orchestrator/v2/instancetypes/{name}/instances/{id}

Accepts
   application/JSON

Content-Type
   application/JSON

Normal Response Codes
   200

Error Response Codes
   401 unauthorized
   404 not found
   500 internal server error

Request
   Resource Instance Request (partial)

Response
   Resource Instance Request
Authorization
Instance provider dependent. Generic Provider: Access Control Link with modify set to true for given domain, project and role you are working on.

DELETE: Deletes an instance of a given type.

URL pattern: /orchestrator/v2/instancetypes/{name}/instances/{id}

Accepts
*/*

Content-Type
application/JSON

Normal response Codes
204

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Authorization
Instance provider dependent. Generic Provider: Access Control Link with modify set to true for given domain, project and role of the user.

GET: Lists actions defined on a given instance of a given type.

URL pattern: /orchestrator/v2/instancetypes/{name}/instances/{id}/services

Accepts
*/*

Content-Type
application/JSON

Normal Response Codes
200

Error Response Codes
401 unauthorized
404 not found
500 internal server error

Authorization
Instance provider dependent. Generic Provider: Access Control Link with view set to true on the instance and services for given domain, project and role of the user.

POST: Launch given action on a given instance of a given type.

URL pattern: /orchestrator/v2/instancetypes/{name}/instances/{id}/services/{serviceid}/launch

Content-Type
application/JSON

Normal Response Codes
202 accepted
Error Response Codes
401 unauthorized
404 not found
500 internal server error

Authorization
Instance provider dependent and ACL with use set to true on the services for given domain, project and role of the user. Access Control Link with use set to true on the instance and services for given domain, project and role of the user.

Resource Instance Providers:
Openstack, catalog and generic providers

Openstack Provider:

Domain Provider:
This provider lists OpenStack keystone domains.

Instance Type
domain

Provider Class
com.ibm.orchestrator.core.instance.providers.openstack.OpenstackDomainProvider

Table 106.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>asc/desc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>Name of the domain</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the domain</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the domain</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from Openstack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enabled</td>
<td>Boolean</td>
<td>Domain enablement status</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>domain</td>
<td>String</td>
<td>Domain id</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Group Provider:**

This provider lists OpenStack keystone groups.

**Instance Type**

group

**Provider Class**

`com.ibm.orchestrator.core.instance.providers.openstack.OpenstackGroupProvider`

**Table 107.**

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>ascii/desc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>name of the group</td>
<td>y</td>
<td>ascii/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the group</td>
<td>y</td>
<td>ascii/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the group</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from Openstack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>domain</td>
<td>String</td>
<td>Domain id</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Heatstack Provider:**

This provider lists OpenStack heat stacks.

**Instance Type**

heat

**Provider Class**

`com.ibm.orchestrator.core.instance.providers.openstack.OpenstackHeatStackProvider`

**Table 108.**

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>ascii/desc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>Name of the server</td>
<td>y</td>
<td>ascii/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the server</td>
<td>y</td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 108. (continued)

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the server</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from OpenStack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>String</td>
<td>server status in OpenStack</td>
<td>y</td>
<td>asc/desc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>openstackId</td>
<td>String</td>
<td>server id in OpenStack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>region</td>
<td>String</td>
<td>OpenStack Region</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>updated</td>
<td>String</td>
<td>Time of last Update</td>
<td>y</td>
<td>asc/desc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>String</td>
<td>Creation Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Project Provider:**

This provider lists OpenStack keystone projects.

**Instance Type**

**project**

**Provider Class**

com.ibm.orchestrator.core.instance.providers.openstack.OpenstackProjectProvider

Table 109.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td></td>
<td>asc/desc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>Name of the project</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the project</td>
<td>y</td>
<td>asc/desc</td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the project</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from OpenStack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 109. (continued)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabled</td>
<td>Boolean</td>
<td>Project enablement status</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>domain</td>
<td>String</td>
<td>Domain id</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User provider:

This provider lists OpenStack keystone users.

Instance Type

user

Provider Class

com.ibm.orchestrator.core.instance.providers.openstack.OpenstackUserProvider

Table 110.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>asc/desc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>Name of the user</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the user</td>
<td></td>
<td></td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the user</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from OpenStack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>enabled</td>
<td>Boolean</td>
<td>User enablement status</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>defaultProject</td>
<td>String</td>
<td>Default project for this user</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>String</td>
<td>Email address of this user</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>domain</td>
<td>String</td>
<td>Domain id</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**VM Provider:**

This provider lists OpenStack nova virtual servers.

**Instance Type**

openstackvms

**Provider Class**

`com.ibm.orchestrator.core.instance.providers.openstack.OpenstackVMProvider`

### Table 111.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>asc/desc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>Name of the server</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the server</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the server</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from OpenStack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>openstackId</td>
<td>String</td>
<td>server id in OpenStack</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>String</td>
<td>server status in OpenStack</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>region</td>
<td>String</td>
<td>OpenStack Region</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>updated</td>
<td>String</td>
<td>Time of last Update</td>
<td>y</td>
<td>asc/desc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>created</td>
<td>String</td>
<td>Creation Time</td>
<td>asc/desc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>keyPair</td>
<td>String</td>
<td>ssh keypair</td>
<td>y</td>
<td>asc/desc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>patternInstanceType</td>
<td>String</td>
<td>instance type of the pattern instance the server belongs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>patternName</td>
<td>String</td>
<td>name of the pattern instance the server belongs</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 111. (continued)

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>patternURI</td>
<td>String</td>
<td>URI to display details of the pattern instance the server belongs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ipAddresses</td>
<td>String</td>
<td>IP Addresses assigned to the server</td>
<td>y</td>
<td>asc/desc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Self-Service Catalog Providers:**

**Offering provider:**

This provider lists self-service catalog offerings.

**Instance Type**

**offering**

**Provider Class**

com.ibm.orchestrator.core.instance.providers.catalog.CatalogOfferingProvider

Table 112.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>name of the offering</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the offering</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>Offering icon</td>
<td></td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the offering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from Catalog</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>Type of this service</td>
<td>asc/desc</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 112. (continued)

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>category</td>
<td>String</td>
<td>Category of this offering</td>
<td>y</td>
<td>asc/desc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Action Provider:*

This provider lists instance actions. Actions are services that can be executed on one or more selected instances.

**Instance Type**

**Offering**

**Provider Class**

com.ibm.orchestrator.core.instance.providers.catalog.CatalogOfferingProvider

Table 113.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>name of the action</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the action</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>Action icon</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the action</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from Catalog</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>String</td>
<td>Type of this service</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>category</td>
<td>String</td>
<td>Category of this offering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>instancetype</td>
<td>String</td>
<td>Type of instance upon which this action can be executed.</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
</tr>
</tbody>
</table>
Table 113. (continued)

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagsAsString</td>
<td>String</td>
<td>Tags combined to a single string.</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Catalog Category Provider:

This provider lists self-service catalog categories. Categories may contain one or more offerings.

Instance Type
category

Provider Class
com.ibm.orchestrator.core.instance.providers.catalogCatalogCategoryProvider

Table 114.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>name of the category</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the category</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>Category icon</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the category</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON result from Catalog</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Generic Provider:

This provider lists generic resources. The provider may be registered multiple times under different instance types.

**Instance Type**

<not registered by default>

**Provider Class**

com.ibm.orchestrator.core.instance.providers.generic.GenericProvider

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Description</th>
<th>Displayed in UI</th>
<th>Sortable</th>
<th>Filterable</th>
<th>Searchable</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>String</td>
<td>Instance Id</td>
<td>asc/desc</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>displayname</td>
<td>String</td>
<td>name of the instance</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the instance</td>
<td>y</td>
<td>asc/desc</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>icon</td>
<td>String</td>
<td>instance icon</td>
<td></td>
<td>asc/desc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>detailsURL</td>
<td>String</td>
<td>URI to display details of the instance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parm</td>
<td>String</td>
<td>JSON object used to store additional information</td>
<td></td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>String</td>
<td>tags to control action availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Task Engine REST API V2**

**JSON Formats**

**Task Response**

```json
{
    "updated_iso" : "2014-03-31T13:05:14+0200",
    "description_message" : "The process is complete.",
    "domain" : "Default",
    "created" : 1396263830875,
    "error" : {
        "resourceBundle" : "com.ibm.orchestrator.messages.orchestratormessages",
        "message" : null,
        "messageKey" : "BPM_PROCESS_COMPLETE",
        "args" : [
            "3"
        ]
    },
    "user" : "ksadmin",
    "parm" : {
        ...
    }
}
```
"created_iso" : "2014-03-31T13:03:50+0200",
"status_localized" : "Completed",
"error_message" : "CTJC00002I: Business process instance 3 completed successfully.",
"status" : "COMPLETED",
"eventTopic" : "com/ibm/orchestrator/serviceinstance/plan/ibm_bpm_process",
"delayInSeconds" : 30,
"project" : "admin",
"id" : "1003",
"updated" : 1396263914896,
"description" : {
   "resourceBundle" : "com.ibm.orchestrator.messages.orchestratormessages",
   "message" : null,
   "messageKey" : "PROCESS_COMPLETE",
   "args" :
   [ ]
}

Task Response

[ Task Response 1,......, Task Response n]

GET: Get all tasks

URL method
/orchestration/v2/tasks

Accepts
*/*

Content-Type
application/JSON

Normal response Codes
200

Error Response Codes
500 internal server error

Request Parameters
Expand: if set to serviceName the information service instance referred by the attribute serviceInstanceId will get returned in the Task Response in the parameter serviceInstance.

Response
Tasks Response

Authorization
No authorization needed but the output is restricted to tasks from all users within the current project. Users with role admin can see all the tasks.

POST: Get all tasks

URL method
/orchestration/v2/tasks

Accepts
*/*

Content-Type
application/JSON

Normal Response Codes
201 created
Error Response Codes
  401 unauthorized
  500 internal server error

Response
  Tasks Response

Authorization
  role: admin

GET: Get the task with a given id

URL method
  /orchestration/v2/tasks/{id}

Accepts
  */*

Content-Type
  application/JSON

Normal Response Codes
  200

Error Response Codes
  404 not found
  500 internal server error

Response
  task Response

Authorization
  No authorization needed but output restricted to role

PUT: Get the task with a given id

URL pattern
  /orchestration/v2/tasks/{id}

Accepts
  */*

Content-Type
  application/JSON

Normal Response Codes
  200

Error Response Codes
  401 unauthorized
  404 not found
  500 internal server error

Response
  Task Response

Authorization
  role: admin or in same project as task

DELETE: Delete the task with a given id
Configuration Providers REST API

Three REST APIs manage the entities under Administration. A REST API consumer can manage the entities in two ways.

- **Manage the entities directly**
  - To manage the entities using the core REST API's, you must use the core REST APIs or OpenStack APIs. If you directly manage entities like domains, users, groups, projects, quotas, you must call the OpenStack APIs. For categories, offerings and actions you must call the core REST APIs. In both cases, the actions will not be run and you must handle all the dependencies.

- **Process flows**
  - If you leverage the logic of the BPM processes as described above, then you need to launch the actions through the core REST API. This REST API is designed for external usage as it involves the business logic that is implemented by the provider. For example, if the REST API provider decides to add an approval process to the *Modify Quota* action of a project, then the provider forces the REST API consumer to launch the action in order to get the approval logic applied.

Managing entities using Core REST APIs:

Before you begin

The mapping in table 1 shows which REST APIs are used to manage the entities.

<table>
<thead>
<tr>
<th>Entity</th>
<th>REST API</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>OpenStack Keystone API</td>
<td>v3/domains</td>
</tr>
<tr>
<td>Project</td>
<td>OpenStack Keystone API</td>
<td>v3/projects</td>
</tr>
<tr>
<td>User</td>
<td>OpenStack Keystone API</td>
<td>v3/users</td>
</tr>
<tr>
<td>Group</td>
<td>OpenStack Keystone API</td>
<td>v3/group</td>
</tr>
<tr>
<td>Quota</td>
<td>OpenStack Compute API</td>
<td>v2.0/{tenant_id}/os-quota-sets</td>
</tr>
</tbody>
</table>
Managing entities using actions:

Any action as described in the Managing Entities using Core Services REST APIs documentation can be invoked per API. The action must be launched in the Core Services REST API. The following procedure is an example of how to launch the Edit Project action on a project via API.

Procedure
1. Get the project provider and the URL for its instances in the instances attribute of the response.

   HTTP method:
   GET

   Example:
   https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project

   ```json
   {  
     "item": {  
       "provider": "com.ibm.orchestrator.core.instance.providers.openstack.OpenstackProjectProvider",
       "detailsview": {  
         "application": "SCOMT",
         "humanservice": "Show Project Details"
       },
       "keyfields": [
         [  
           "instanceattribute": "displayname",
           "header": "Name"
         ],
         [  
           "instanceattribute": "description",
           "header": "Description"
         ],
         [  
           "instanceattribute": "enabled",
           "header": "Enabled?"
         ]
       ],
       "tags": [  
         "enabled",
         "disabled"
       ],
       "icon": "Web Icon:glyphicons_232_cloud",
       "type": "admin",
       "name": "project",
       "description": "Show your OpenStack projects."
     },
     "instances": {  
       "href": "https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project"
     }
   }
   ```

<table>
<thead>
<tr>
<th>Entity</th>
<th>REST API</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>Core REST API</td>
<td>orchestrator/v2/categories</td>
</tr>
<tr>
<td>Offering</td>
<td>Core REST API</td>
<td>orchestrator/v2/offerings</td>
</tr>
<tr>
<td>Action</td>
<td>Core REST API</td>
<td>orchestrator/v2/offerings</td>
</tr>
</tbody>
</table>
2. Get the instance you want to manage and find the ID and name. You can also use the API filter to search the name attribute.

**HTTP Method:**

GET

**Example:**

https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/instances

```json
{
    "start": 0,
    "limit": 10,
    "total": 4,
    "first": {
        "href": "https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/instances?_limit=10&_start=0"
    },
    "previous": null,
    "next": null,
    "last": {
        "href": "https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/instances?_limit=10&_start=0"
    },
    "items": [
        {
            "href": "https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/instances/4ae7ade7e4724c69ab90246ea7965e6",
            "item": {
                "enabled": true,
                "domain": "4ae7ade7e4724c69ab90246ea7965e6",
                "tags": ["enabled"],
                "icon": null,
                "id": "4ae7ade7e4724c69ab90246ea7965e6",
                "parm": {
                    "enabled": true,
                    "domain_id": "default",
                    "links": {
                        "self": "http://9.152.137.35:5000/v3/projects/4ae7ade7e4724c69ab90246ea7965e6",
                        "id": "4ae7ade7e4724c69ab90246ea7965e6",
                        "name": "admin",
                        "description": "admin Tenant"
                    },
                    "description": "admin Tenant",
                    "detailsURL": "https://xvm127.boeblingen.de.ibm.com:8443/teamworks/executeServiceByName?processApp=SCOMT&serviceName=Show+Project+Details&tw.local.projectId=4ae7ade7e4724c69ab90246ea7965e6&tw.local.domainId=default&tw.local.authUser=admin&tw.local.authDomain=
```
3. Get the actions that are applicable to projects. Get the link to in the services attribute of the response in step 1. Get the services and find the Edit Project action by name and remember its ID.

**HTTP method:**
GET

**Example:**
https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/services

```
...{
  "href": "https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/services/69",
  "item": {
    "human_service": "Edit Project Action",
    "priority": 0,
    "human_service_app_name": "SCOrchestrator Multi-Tenancy Toolkit",
    "implementation_type": null,
    "created": 1401827772,
    "human_service_app_short_name": "SCOMT",
    "process_app_id": "2066.227c57b3-a5e5-4e5b-a283-c920cf9bed50",
    "acl": [
      ...
    ],
    "name": "Edit Project",
    "ownerid": 0,
    "instancetype": "project",
    "process": "Edit Project Action",
    "operation_type": "singleInstanceAction",
    "human_service_app_short_name": "SCOMT",
    "process_app_id": "2066.227c57b3-a5e5-4e5b-a283-c920cf9bed50",
    "tags": [
      "enabled"
    ],
    "process_app_name": "SCOrchestrator Multi-Tenancy Toolkit",
    "icon": "act16_return",
    "updated": 1401827772,
    "id": 69,
    "process_app_short_name": "SCOMT",
    "description": "Edit the project details",
    "category": 31
  }
},
...```

4. Launch the action with the ID from step 3, passing the ID of the selected instance (project) from step 2 in the request body. The call returns a task which is in the state **NEW** and a new ID.

**Note:** For all actions of type "createInstance", the ID of the domain must be passed in the "instances" array of the PUT request.

**HTTP method:**
POST

**Body:**
Example:
https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/services/69/launch

{
  "updated_iso": "1970-01-01T00:00:00+0100",
  "description_message": "HS_OFFERING_INVOCATION",
  "domain": "Default",
  "message": "Launched",
  "created": 1402569924045,
  "error": null,
  "user": "admin",
  "parm": {
    "plan": {
      "human_service": "Edit Project Action",
      "priority": 0,
      "human_service_app_name": "SCOrchestrator Multi-Tenancy Toolkit",
      "implementation_type": null,
      "created": 1401827772,
      "human_service_app_short_name": "SCOMT",
      "process_app_id": "2066.227c57b3-a5e5-4e5b-a283-c920cf9bed50",
      "acl": []
    },
    "name": "Edit Project",
    "ownerid": 0,
    "instancetype": "project",
    "process": "Edit Project Action",
    "operation_type": "singleInstanceAction",
    "human_service_app_id": "2066.227c57b3-a5e5-4e5b-a283-c920cf9bed50",
    "tags": [
      "enabled"
    ],
    "process_app_name": "SCOrchestrator Multi-Tenancy Toolkit",
    "icon": "act16_return",
    "updated": 1401827772,
    "id": 69,
    "process_app_short_name": "SCOMT",
    "description": "Edit the project details",
    "category": 31
  },
  "instances": [
    "default"
  ]
}

"created_iso": "2014-06-12T12:45:24+0200",
"status_localized": "New",
"error_message": null,
"status": "NEW",
"eventTopic": "com/ibm/orchestrator/serviceinstance/plan/ibm_bpm_process",
"delayInSeconds": 0,
"project": "admin",
"id": "1521",
"updated": 0,
"redirect": "/teamworks/executeServiceByName?processApp=SCOMT&serviceName=Edit+Project+Action&tw.local.operationContextId=1521",
"description": {
  "resourceBundle": "com.ibm.orchestrator.messages.orchestratormessages",
  "message": "HS_OFFERING_INVOCATION",
}
5. Set the parameters of the task which are the input for the action. Then set the status of the task to QUEUED to queue the task for execution. In the body the description is set to test and the other attributes remain the same.

**HTTP method:**
PUT

**Body:**

```json
{
    "status": "QUEUED",
    "parm": {
        "OperationParameter": "<variable type="Project">
            
            <name type="String">admin</name>
            
            <description type="String">test</description>
            
            <enabled type="Boolean">true</enabled>
            
            <id type="String">4ae7ade7e4724c69ab90246ea72965e6</id>
            
            <domainId type="String">default</domainId>
        </variable>
    }
}
```

**Example:**

6. Check if the task succeeded or failed. The status will switch to RUNNING. If the task succeeds the status says COMPLETED. If the task fails the status says FAILED and an error_message is shown. In the example, the process completed.

**HTTP method:**
GET

**Example:**

```json
{
    "error_message": "CTJC000021: Business process instance 79 completed successfully.",
    "status": "COMPLETED",
}
```

7. Verify if the action applied the changes on the entity. Finally, it is possible to ensure if the change happened on the instance.

**HTTP method:**
GET

**Example:**
https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/instances/4ae7ade7e4724c69ab90246ea72965e6

```json
{
    "href": "https://xvm127.boeblingen.de.ibm.com:8443/orchestrator/v2/instancetypes/project/instances/4ae7ade7e4724c69ab90246ea72965e6",
    "item": {
        "enabled": true,
        "domain": "4ae7ade7e4724c69ab90246ea72965e6",
        "tags": [
            "enabled"
        ],
        "icon": null,
    }
}
```
Core REST API for compatibility with earlier versions

The REST APIs described in the following sections were replaced in IBM Cloud Orchestrator 2.4 but they are still valid for compatibility with SmartCloud Orchestrator 2.3.

Self-service offering REST API:

You can use this set of REST API calls to interact with the self-service offerings in IBM Cloud Orchestrator.

Create a self-service offering:

Use this REST call to create a self-service offering.

Available HTTP method

Table 116. Create a self-service offering REST API call

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/services">https://hostname/resources/services</a></td>
</tr>
</tbody>
</table>
Table 116. Create a self-service offering REST API call (continued)

<table>
<thead>
<tr>
<th>Response</th>
<th>The response of the server contains the specified offering. It has the following set of attributes:</th>
</tr>
</thead>
</table>

| Return values | • 201 Returns the created service or offering  
• 500 Internal service error |

An entry has the following attributes:

- **category** - optional, the category to which the offering belongs.
- **created** - the creation time of the self-service offering, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.
- **description** - optional, a short description of the offering.
- **human_service** - optional, the URL of a IBM Business Process Manager human service (coach), a user interface to provide user input.
- **human_service_app_id** - optional, depending on the human_service attribute, the ID of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_name** - optional, depends on if the human_service attribute is set, the name of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_short_name** - the short name of the IBM Business Process Manager human service application.
- **icon** - optional, an offering can have an icon assigned that is displayed inside the Self-Service Catalog.
- **id** - the ID of the offering.
- **implementation_type** - the two possible values are 'ibm_bpm_process' and 'script'.
- **name** - the name of the offering.
- **operation_type** - the only possible value is "service".
- **ownerid** - the owner of the user who triggered the offering.
- **process** - the name of the IBM Business Process Manager process bound to the offering.
• **process_app_id** - the ID of the IBM Business Process Manager application in which the process is defined.

• **process_app_name** - the name of the IBM Business Process Manager application in which the process is defined.

• **process_app_short_name** - the short name of the IBM Business Process Manager process application.

• **updated** - the time when the self-service offering was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

The following listing shows an example response that can be retrieved by way of the request:

```
{
   "human_service": "Sample_ReportProblem",
   "human_service_app_name": "SCOrchestrator_Toolkit",
   "implementation_type": "ibm_bpm_process",
   "human_service_app_short_name": "SCOTLKT",
   "process_app_id": "2066.596706e1-2e92-4fb1-a2dd-e0e4bdc4f7fc",
   "name": "Problem report",
   "created": 1242965374865,
   "updated": 1242965392870,
   "ownerid": 2,
   "process": "Sample_Report",
   "operation_type": "service",
   "human_service_app_id": "2066.596706e1-2e92-4fb1-a2dd-e0e4bdc4f7fc",
   "process_app_name": "SCOrchestrator_Toolkit",
   "icon": "Configuration Icon:ge100_config_24",
   "id": 5,
   "process_app_short_name": "SCOTLKT",
   "description": "Report a problem",
   "category": ""
}
```

List all self-service offerings:

Use this REST API method to list all self-service offerings.

**Available HTTP method**

*Table 117. Get list of all self-service offerings*

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/services">https://hostname/resources/services</a></td>
</tr>
<tr>
<td>Response</td>
<td>The response of the server contains a list of available self-service offerings. Each offering has the following set of attributes:</td>
</tr>
</tbody>
</table>

```json
{
   category:
   created:
   description:
   human_service:
   human_service_app_id:
   icon:
   id:
   implementation_type:
   name:
   operation_type:
   ownerid:
   process:
   process_app_id:
   updated:
}
```
An entry has the following attributes:

- **category** - optional, a category to which the offering belongs.
- **created** - the creation time of the self-service offering, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.
- **description** - optional, a short description for the offering.
- **human_service** - optional, the URL of a IBM Business Process Manager human service (coach), a user interface to provide user input.
- **human_service_app_id** - optional, depending on the human_service attribute. The ID of the IBM Business Process Manager application to which the human_service belongs.
- **icon** - optional, an offering can have an icon assigned that will be displayed inside the Self-Service Catalog.
- **id** - ID of the offering.
- **implementation_type** - possible values are 'ibm_bpm_process' or 'script'.
- **name** - the name of the offering.
- **operation_type** - the only possible value is 'service'.
- **ownerid** - the owner of the user who triggered the offering.
- **process** - the name of the IBM Business Process Manager process bound to the offering.
- **process_app_id** - the ID of the IBM Business Process Manager application in which a process is defined.
- **updated** - the time when the self-service offering was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

The following listing shows an example response that can be retrieved by way of the request:

```json
[
  {
    "human_service": "Sample_ReportProblem",
    "implementation_type": "ibm_bpm_process",
    "process_app_id": "2066.596706e1-2e92-4fb1-a2dd-e0e4bdc4f7fc",
    "name": "Problem report",
    "created": 1242965374865,
    "updated": 1242965392870,
    "ownerid": 2,
    "process": "Sample_Report",
    "operation_type": "service",
    "human_service_app_id": "2066.596706e1-2e92-4fb1-a2dd-e0e4bdc4f7fc",
    "icon": "Job Icon:ge100_job_24",
    "id": 5,
    "description": "Report a problem",
    "category": 5
  }
]
```
Get entries for a specific self-service offering:

Use this REST call to retrieve information about a self-service offering with a specified ID.

Available HTTP method

| Table 118. Get entries for a specific self-service offering REST API call |
|-----------------------------|-----------------------------|
| **HTTP method**             | GET                         |
| **URL pattern**             | https://hostname/resources/services/{id}?acl=true                   |

Response

The response of the server contains the specified offering. It has the following set of attributes:

```plaintext
{
  acl:
  category:
  created:
  description:
  human_service:
  human_service_app_id:
  human_service_app_name:
  human_service_app_short_name:
  icon:
  id:
  implementation_type:
  name:
  operation_type:
  ownerid:
  process:
  process_app_id:
  process_app_name:
  process_app_short_name:
  updated:
}
```

**Note:** The acl attribute is only returned when the optional query parameter acl is passed with the value true.

Return values

- 200 Returns the service or offering associated with the given ID
- 403 If the client is not on the offering's ACL, they are not authorized to perform this action.
- 404 No offering exists with the given ID
- 500 Internal server error

An entry has the following attributes:

- **category** - optional, the category to which the offering belongs.
- **created** - the creation time of the self-service offering, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.
- **description** - optional, a short description of the offering.
- **human_service** - optional, the URL of an IBM Business Process Manager human service (coach), a user interface to provide user input.
- **human_service_app_id** - optional, depending on the human_service attribute, the ID of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_name** - optional, depends on if the human_service attribute is set, the name of the IBM Business Process Manager application to which the human service belongs.

- **human_service_app_short_name** - the short name of the IBM Business Process Manager human service application.

- **icon** - optional, an offering can have an icon assigned that is displayed inside the Self-Service Catalog.

- **id** - the ID of the offering.

- **implementation_type** - the two possible values are 'ibm_bpm_process' and 'script'.

- **name** - the name of the offering.

- **operation_type** - the only possible value is "service".

- **ownerid** - the owner of the user who triggered the offering.

- **process** - the name of the IBM Business Process Manager process bound to the offering.

- **process_app_id** - the ID of the IBM Business Process Manager application in which the process is defined.

- **process_app_name** - the name of the IBM Business Process Manager application in which the process is defined.

- **process_app_short_name** - the short name of the IBM Business Process Manager process application.

- **updated** - the time when the self-service offering was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

The following listing shows an example response that can be retrieved by way of the request:

```json
{
  "human_service": "Sample_ReportProblem",
  "human_service_app_name": "SCOrchestrator_Toolkit",
  "implementation_type": "ibm_bpm_process",
  "human_service_app_short_name": "SCOTLKT",
  "process_app_id": "2066.596706e1-2e92-4fb1-a2dd-e0e4bdc4f7fc",
  "name": "Problem report",
  "created": 1242965374865,
  "updated": 1242965392870,
  "ownerid": 2,
  "process": "Sample_Report",
  "operation_type": "service",
  "human_service_app_id": "2066.596706e1-2e92-4fb1-a2dd-e0e4bdc4f7fc",
  "process_app_name": "SCOrchestrator_Toolkit",
  "icon": "Configuration Icon:ge100_config_24",
  "id": 5,
  "process_app_short_name": "SCOTLKT",
  "description": "Report a problem",
  "category": 5
}
```

"acl":

[ ]
Delete a specific self-service offering:

Use this REST API call to delete a specific self-service offering.

Available HTTP method

Table 119. Delete a self-service offering REST API call

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>DELETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/services/%7Bid%7D">https://hostname/resources/services/{id}</a></td>
</tr>
<tr>
<td>Response</td>
<td>The self-service offering is deleted.</td>
</tr>
</tbody>
</table>
| Return values | • 204 Deletes an offering and its ACL  
• 401 The client is not authorized to perform this action as they are not on the offering's ACL  
• 404 No offering has the given ID.  
• 500 Internal server error |

Update a specific self-service offering:

Use this REST API call to update a specific self-service offering.

Available HTTP method

Table 120. Update a self-service offering REST API call

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>PUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td><a href="https://hostname/resources/services/%7Bid%7D">https://hostname/resources/services/{id}</a></td>
</tr>
<tr>
<td>Response</td>
<td>The self-service offering is updated.</td>
</tr>
</tbody>
</table>
| Return values | • 201 - Updates an existing offering  
• 400 - Decode failure. Request body does not contain valid JSON  
• 401 - Authorization failure  
• 404 - Update failure  
• 500 - Internal server error |

Structure of request body:

```json
{
    "<attribute>": "<attribute_value>
}
```

The following listing shows sample content of request body to update the description of a self-service offering:

```json
{
    "description": "Changed description"
}
```
Executing a self-service offering:

Three different scenarios are supported.

Executing a self-service offering which does not have any parameters

In this scenario only one REST-call is required:
REST POST /resources/automation/<offering-id>

(IDs for the offerings can be retrieved with REST GET /resources/services).

Executing self-service offering without "human service" providing all parameters

1. Run REST POST /resources/automation/<offering-id>
2. Retrieve the <request-id> from the response.
3. Run REST PUT /kernel/tasks/<request-id> with required parameters as payload.

Executing self-service offering with "human service" to collect all the required parameters through the UI

1. Run REST POST /resources/automation/<offering-id>
2. Get <human-service-url> from the response.
3. Use the URL in an iframe, for example, to display the user interface and collect parameters (when you click Submit, the business process is kicked off).

Step 1 - Start the request by creating the operation context

Use the IBM Cloud Orchestrator interface to create the operationContext by posting to the offering API. Note the mandatory API version and content-type headers, and the empty json object {} we send over, an offering with ID equal to 31 is used in this example:

This is all that needs to be done for scenario A. For scenario B, search for the request ID in the response of the call:
{..."id": "1004", ...}

or in case of scenario C (delegated UI), search for the human service URL:
{.... "redirect": "/teamworks/executeServiceByName?processApp=SCOTLKT&serviceName=Sample_UserInterface&tw.local.operationContextId=1004",....}

For scenario C, use the URL in an iframe, for example, to display the user interface and collect parameters (when you click Submit, the business process is kicked off).

Step 2 - Trigger the execution of the offering workflow (Business Process Manager process)

Step 2 is only required for scenario B. This step passes the parameters to Business Process Manager process and starts the process, use the request ID from the previous step.
Sample call: this call passes the values Hello and Phone for a
Sample_BusinessObject consisting of two string attributes (field1 and field2):
curl -ku admin:passw0rd -X PUT -d '{"status":"QUEUED","parm":{"OperationParameter":
"<variable type="Sample_BusinessObject">"

"<field1 type="String">"<![CDATA[Hello]]>"</field1>

"<field2 type="String">"<![CDATA[Phone]]>"</field2> <"variable"}'}
-H "Content-Type: application/json" https://10.102.98.11/kernel/tasks/1004

where status must be set to QUEUED so that the process is started, and parm must
contain the input values for the process – the values must be passed in the
serialized form of the related Business Process Manager business object as returned
by the Business Process Manager
tw.system.serializer.toXML(tw.local.inputParameterObject)

How to get details about the input parameter datatype

In case you need to programmatically introspect input parameters of the offering
workflow the following things can be performed:

- Use REST interface of IBM Cloud Orchestrator to retrieve details about a
  registered offering:
  GET .../resources/services/{id}

Among the data there are:

process
The name of the IBM Business Process Manager process bound to the
offering.

process_app_id
The ID of the IBM Business Process Manager application in which the
process is defined.

For details, refer to "Get entries for a specific self-service offering” on page 717.

- Use REST Interface for BPD-related resources to get details about exposed items:
  GET .../rest/bpm/wle/v1/exposed

Find the process with display=process and processAppID=process_app_id.
{"status":"200", "data":[ "exposedItemsList":[ { "type":"process", 
"itemID":"25.8403dd37-e049-46f5-8952-b7a46f9d198f", 
"processAppID":"2066.931b0053-02bd-4f47-ac72-4eb527467383", 
"snapshotID":"2064.73dd1d1a-b533-46e9-ba79-c94cb3b0de87", 
"snapshotName":"version 2.8.1", 
"display":"HR Open New Position", "ID":"2015.204" }

For details, refer to http://pic.dhe.ibm.com/infocenter/dmndhelp/v8r5m0/
index.jsp?topic=%2Fcom.ibm.wbpm.ref.doc%2Frest%2Fbpmrest%
%2Frest_bpm_wle_v1_exposed.htm

- Use REST Interface for BPD-related resources to get details about the process
  model:
  GET .../rest/bpm/wle/v1/processModel/{bpdId}?processAppId={string}&parts=dataModel

You may want to use "parts=all" which will then return more information
about the process such as detailed descriptions. For the bpdId use itemID from
above, for processAppId use the one above:

{ "status":"200", "data":{ ... }, "DataModel":{ ... } }

Chapter 10. Reference 721
Sample Output:

Note that only the bold sections in the following output excerpt are of interest:

- Getting the type of the inputParameterObject
- Getting the type detailed information, here the parameters to that service are two string parameters named field1 and field2

```json
{
  "status": "200",
  "data": {
    "DataModel": {
      "properties": {
        "message": { "type": "String", "isList": false },
        "returnFromRest": { "type": "String", "isList": false }
      },
      "inputs": {
        "operationContext": { "type": "OperationContext", "isList": false },
        "inputParameterObject": { "type": "Sample_BusinessObject", "isList": false }
      },
      "Sample_BusinessObject": {
        "properties": {
          "field1": { "isList": false, "type": "String" },
          "field2": { "isList": false, "type": "String" }
        },
        "type": "object",
        "ID": "12.2c079fa7-89a0-426c-a3c1-079be08930ac",
        "isShared": false
      }
    }
  }
}
```

How to get an example for the input parameter payload

In case you need to get an example of the input parameter payload, you can trigger a request through the IBM Cloud Orchestrator and query the input parameters of the request (it could be still running or even finished already).

Therefore you must:
1. Retrieve the request ID.
2. Retrieve the input parameters for that request.

To get the request ID, select the triggered request in the REQUEST HISTORY view and note the request ID as listed in the address bar and as part of the request details on the right portion of the panel.

Now using the REST call to retrieve details about the request use:

```
GET .../kernel/tasks/{request-id}
```

for example:
```
curl -ku admin:passw0rd -H "X-IBM-Workload-Deployer-API-Version: 3.1"
```

Within the response, search for Operation Parameter. Sample excerpt:
```
"OperationParameter": ":<variable type="\MyRequest\" 
<vpmoNumber type="\Integer\"><![CDATA[116560]]></vpmoNumber> 
<appId type="\Integer\"><![CDATA[19073]]></appId>"
How to get information about a request

Once a request is started the IBM Cloud Orchestrator system can be queried for the actual status of that request. This is done using the REST call:

GET /kernel/tasks/{id}

For example:


Among other information, the response contains information about the STATUS of the request. For details on possible values, refer to "Get entries for a specific task" on page 734.

Sample response (excerpt) from REST GET /kernel/tasks/{id}:

```json
{
  "updated_iso": "2014-02-19T17:54:15+0100",
  "description_message": "PROCESS_COMPLETE",
  "domain": "Default",
  "created": 1392828461580,
  "error": { ... },
  "serviceInstance": {
    "virtualMachines": [{
      "memory": 4096,
      "hypervisorid": "/resources/hypervisors/PM-1",
      "hostname": "SC-192-168-0-103.RegionOne.example.com",
    },
    ],
    "user": "admin",
    "parm": {
      "startPlanByPlugpointEventHandler": "done",
      "CUSTOM_PARM1": "abc",
      "CUSTOM_PARM2": "xyz",
      "OperationParameter": "<variable ... /></variable>",
      "serviceInstanceId": "282",
      "plan": [ ... ],
      "processId": "1356"
    },
    "created_iso": "2014-02-19T17:47:41+0100",
    "statusLocalizedMessage": "TASKSTATUS_COMPLETED",
    "error_message": "BPM_PROCESS_COMPLETE",
    "status": "COMPLETED",
    "eventTopic": "com:/ibm/orchestrator/serviceinstance\plan\ibm_bpm_process",
    "delayInSeconds": 30,
    "project": "admin",
    ...
  }
}
```

Once the request has finished, there may be the need to retrieve further information about what has been done by this request. If this is needed, the
Business Process Manager process can use certain building blocks (Integration Services) to store process specific information in the request (actually in the operation context).

For details, see Integration Services.

The two integration services of interest here are SetOperationContextParameters and SetServiceInstanceId.

SetOperationContextParameters enables the Process Designer to easily store any custom specific set of key/value pairs in the operation context object. These parameters are then part of the response of the GET /kernel/tasks/{id} REST call. The key/value pairs will be added to the parm section in the response. This capability can be used to either transfer data from human service to execution process, or to store information in the operation context which is useful to be retrieved programmatically once the request has finished.

SetServiceInstanceId enables the Process Designer to store the ID of the service instance in the operation context. This is useful if the process is about creating a virtual service instance, and therefore as a result of this process, a reference to the newly provisioned virtual system instance should be stored. The virtual system instance ID as provided by the deploy pattern building block can be used to store as the service instance ID in the operation context. If the operation context contains a valid service instance ID, the REST call GET /kernel/tasks/{id} can be used to get all information of the virtual service instance in the response. Therefore use GET /kernel/tasks/{id}?expand=serviceInstance.

In the sample response above the following lines are of interest:

- If SetServiceInstanceId is used, the following property will be in the response in the parm section:
  "serviceInstanceId" : "282"

- If ?expand=serviceInstance is used for the REST call, the following section will be part of the response:
  "serviceInstance" : {
      "virtualMachines" : [{
         "memory" : 4096,  
         "hypervisorId" : "/resources/hypervisors/PM-1",  
         "hostname" : "SC-192-168-0-103.RegionOne.example.com",
      }]

- If SetOperationContextParameters is used the following two custom properties will show up in the response in the parm section:
  "CUSTOM_PARM1": "abc",  
  "CUSTOM_PARM2": "xyz",

These two custom parameters have been added to the key/value map in the business process by using the following Java script code, for example:

```
tw.local.parMap = new tw.object.Map();
tw.local.parMap.put('CUSTOM_PARM1', 'abc');
tw.local.parMap.put('CUSTOM_PARM2', 'xyz');
```

and mapping the local variable parMap as input for SetOperationContextParameters.

**Note:** You only need to set the operation context ID and the parameters; all other input can be left off, the defaults are fine here.
End-To-End example

To execute the Stop Virtual System Instance Self-Service offering:

1. Get the ID value of the Virtual System to stop:

   ```
curl -v -ku admin:passw0rd -H "X-IBM-Workload-Deployer-API-Version: 3.1"
   https://172.17.41.98/resources/virtualSystems -X GET

   ...
   {
       "desiredstatus_text": null,
       "currentstatus_text": "Started",
       "name": "GP_wmbhve",
       "desiredstatus": null,
       "pattern": "/resources/patterns/22",
       "id": 3,
       "currentmessage_text": "Virtual system is ready",
       "created": 1391537634322,
       "currentstatus": "RM01006",
       "creator": "admin",
       "currentmessage": "RM07009",
       "owner": "/resources/users/40",
       "updated": 1391723628540
   }
   ...
   
   2. Get the ID, the process_app_id, and the process value of the offering to execute:

   ```
curl -ku admin:passw0rd -H "X-IBM-Workload-Deployer-API-Version: 3.1"
   -H "Content-Type:application/json" https://172.17.41.98/resources/services -X GET

   ...
   {
       "human_service": "Stop Single vSys Instance",
       "implementation_type": "ibm_bpm_process",
       "created": 1390828290449,
       "process_app_id": "2066.6ecd41b3-6c42-47e4-a69e-117d77f4104e",
       "name": "Stop Virtual System Instance",
       "ownerid": 40,
       "process": "Stop Single vSys Pattern Instance",
       "operation_type": "service",
       "human_service_app_id": "2066.6ecd41b3-6c42-47e4-a69e-117d77f4104e",
       "icon": "Cloud Icon:ge100_virtualfabric_24",
       "updated": 1390828290449,
       "id": 3,
       "description": "",
       "category": "2"
   }
   ...
   
   3. Create the operation context by passing the ID that you got in Step 2 and getting the ID value from the response:

   ```
curl -ku admin:passw0rd -H "X-IBM-Workload-Deployer-API-Version: 3.1"
   -H "Content-Type:application/json" -d '{}'
   https://172.17.41.98/resources/automation/3
   -X POST

   (example):

   {
       "updated_iso": "2014-02-06T16:15:34-0500",
       "description_message": "Starting offering Stop Virtual System Instance. ",
       "domain": "Default",
       "message": "The action was submitted successfully. See the History section of this instance to track the action progress."
   }
   ```
4. Get details about the input parameters:

a. Get the itemId value from the response, searching for the process and the process_app_id values got in Step 2:

```
curl -v -ku admin:passw0rd https://172.17.41.99:9443/rest/bpm/wle/v1/exposed -X GET
```

...  
"type": "process",  
"itemID": "25.f28991e-2884-4009-96d6-581f4bceef980",  
"itemReference": "/25.f28991e-2884-4009-96d6-581f4bceef980",  
"processAppID": "2066.6ecd41b3-6c42-47e4-a69e-117d77f4104e",  
"snapshotID": "2064.f3012098-d3f0-4609-8aff-acf0038c9f78",  
"snapshotName": "2300_20131029",  
"snapshotCreatedOn": "2013-10-29T14:55:24Z",  
"display": "Stop Single vSys Pattern Instance",  
"tip": true,  
"branchID": "2063.44ef08ae-2760-4d2b-ab11-7de7b3475617",  
"branchName": "Main",  
"startURL": "/rest/bpm/wle/v1/process?action=start&bpdId=25.f28991e-2884-4009-96d6-581f4bceef980"
b. Get details about the process model, passing the idemId that you got in step a and the process_app_id that you got in Step 2:

```bash
curl -v -ku admin:passw0rd
?processAppId=2066.6ecd41b3-6c42-47e4-a69e-117d7f4104e&parts=dataModel -X GET
```

(example):

```json
{
  "status": "200",
  ...
  "inputs": {
    "operationContext": {
      "type": "OperationContext",
      "isList": false,
    },
    "InputParameterObject": {
      "type": "VirtualSystem",
      "isList": false
    }
  },
  "VirtualSystem": {
    ...
    "properties": {
      "currentstatus_text": {
        "isList": false,
        "type": "String"
      },
      "envProfileId": {
        "isList": false,
        "type": "String"
      },
      "currentstatus": {
        "isList": false,
        "type": "String"
      },
      "name": {
        "isList": false,
        "type": "String"
      },
      "id": {
        "isList": false,
        "type": "String"
      },
      "currentmessage": {
        "isList": false,
        "type": "String"
      }
    }
  }
}
```

5. Start the execution of the offering, passing in the ID of the Virtual System to stop that you got from Step 1, and the task ID value that you got from Step 3:

```bash
curl -ku admin:passw0rd -X PUT -d '{"status": "QUEUED",
  "parm": {
    "OperationParameter": "<variable type="VirtualSystem">
    <id type="String">4</id>
  </variable>"},
  "plan": {
    "human_service": "Stop Single vSys Instance",
    "priority": 5,
    "human_service_app_name": "SCOrchestrator_Support_vSys_Toolkit",
    "implementation_type": "ibm_bpm_process",
    "human_service_app_short_name": "SCOVSYS",
    "created": 1390828290449,
    "process_app_id": "2066.6ecd41b3-6c42-47e4-a69e-117d7f4104e",
    "name": "Stop Virtual System Instance",
    "ownerid": 40,
    "process": "Stop Single vSys Pattern Instance",
    "operation_type": "service",
    "human_service_app_id": "2066.6ecd41b3-6c42-47e4-a69e-117d7f4104e",
    "icon": "CTJCO113B1ge100_virtualfabric_24",
    "process_app_name": "SCOrchestrator_Support_vSys_Toolkit",
    "event": null,
    "id": 3,
    "updated": 1390828290449,
    "description": "",
    "process_app_short_name": "SCOVSYS",
    "category": "2",
    "apply_to_all_pattern": 0
}'
```

Self-service catalog REST API:

You can use this set of REST API calls to interact with the Self-Service Catalog in IBM Cloud Orchestrator.

<table>
<thead>
<tr>
<th>ID</th>
<th>The unique identifier of the category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the category, that is, the name that appears in the service catalog</td>
</tr>
<tr>
<td>Description</td>
<td>The description of the category</td>
</tr>
<tr>
<td>Icon</td>
<td>The name of the icon that is used to represent the category</td>
</tr>
</tbody>
</table>
Table 121. Categories (continued)

| Isbuiltin | "1 or 0" for "true or false" whether the category is provided by the product |

Create category:

Use this REST call to create a category.

Available HTTP method

Table 122. Create a category REST API call

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td>/resources/automationcategories</td>
</tr>
<tr>
<td>Response</td>
<td>The response of the server contains the specified offering. It has the following set of attributes:</td>
</tr>
</tbody>
</table>

```json
{
    category:
    created:
    description:
    human_service:
    human_service_app_id:
    human_service_app_name:
    human_service_app_short_name:
    icon:
    id:
    implementation_type:
    name:
    operation_type:
    ownerid:
    process:
    process_app_id:
    process_app_name:
    process_app_short_name:
    updated:
}
```

An entry has the following attributes:

- **category** - optional, the category to which the offering belongs.
- **created** - the creation time of the self-service offering, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.
- **description** - optional, a short description of the offering.
- **human_service** - optional, the URL of a IBM Business Process Manager human service (coach), a user interface to provide user input.
- **human_service_app_id** - optional, depending on the human_service attribute, the ID of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_name** - optional, depends on if the human_service attribute is set, the name of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_short_name** - the short name of the IBM Business Process Manager human service application.
- **icon** - optional, an offering can have an icon assigned that is displayed inside the Self-Service Catalog.
• **id** - the ID of the offering.
• **implementation_type** - the two possible values are 'ibm_bpm_process' and 'script'.
• **name** - the name of the offering.
• **operation_type** - the only possible value is "service".
• **ownerid** - the owner of the user who triggered the offering.
• **process** - the name of the IBM Business Process Manager process bound to the offering.
• **process_app_id** - the ID of the IBM Business Process Manager application in which the process is defined.
• **process_app_name** - the name of the IBM Business Process Manager application in which the process is defined.
• **process_app_short_name** - the short name of the IBM Business Process Manager process application.
• **updated** - the time when the self-service offering was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

```json
{
    "isbuiltin": 0,
    "icon": "Web Category Icon:ge100_webcatalog_24",
    "name": "Bucket",
    "description": ""
}
```

**Get the list of categories:**

Use this REST call to get a list of categories.

**Available HTTP method**

*Table 123. Get the list of categories REST API call*

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td>/resources/automationcategories</td>
</tr>
</tbody>
</table>

Response

The response of the server contains the specified offering. It has the following set of attributes:

```json
{
    category:
    created:
    description:
    human_service:
    human_service_app_id:
    human_service_app_name:
    human_service_app_short_name:
    icon:
    id:
    implementation_type:
    name:
    operation_type:
    ownerid:
    process:
    process_app_id:
    process_app_name:
    process_app_short_name:
    updated:
}
```
An entry has the following attributes:

- **category** - optional, the category to which the offering belongs.
- **created** - the creation time of the self-service offering, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.
- **description** - optional, a short description of the offering.
- **human_service** - optional, the URL of an IBM Business Process Manager human service (coach), a user interface to provide user input.
- **human_service_app_id** - optional, depending on the human_service attribute, the ID of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_name** - optional, depends on if the human_service attribute is set, the name of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_short_name** - the short name of the IBM Business Process Manager human service application.
- **icon** - optional, an offering can have an icon assigned that is displayed inside the Self-Service Catalog.
- **id** - the ID of the offering.
- **implementation_type** - the two possible values are 'ibm.bpm_process' and 'script'.
- **name** - the name of the offering.
- **operation_type** - the only possible value is "service".
- **ownerid** - the owner of the user who triggered the offering.
- **process** - the name of the IBM Business Process Manager process bound to the offering.
- **process_app_id** - the ID of the IBM Business Process Manager application in which the process is defined.
- **process_app_name** - the name of the IBM Business Process Manager application in which the process is defined.
- **process_app_short_name** - the short name of the IBM Business Process Manager process application.
- **updated** - the time when the self-service offering was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

```json
{
   "isbuiltin": 0,
   "icon": "Web Category Icon:ge100_webcatalog_24",
   "name": "Bucket",
   "id": 8,
   "description": ""
},
{
   "isbuiltin": 0,
   "icon": "Cloud Category Icon:ge100_virtualfabriccatalog_24",
   "name": "Help Desk",
   "id": 3,
   "description": ""
}
```
Get the details of a single category:

Use this REST call to get the details of a single category.

Available HTTP method

Table 124. Get the details of a single category REST API call

<table>
<thead>
<tr>
<th>HTTP method</th>
<th>GET</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL pattern</td>
<td>/resources/automationcategories/8</td>
</tr>
</tbody>
</table>

Response

The response of the server contains the specified offering. It has the following set of attributes:

```json
{
    category:
    created:
    description:
    human_service:
    human_service_app_id:
    human_service_app_name:
    human_service_app_short_name:
    icon:
    id:
    implementation_type:
    name:
    operation_type:
    ownerid:
    process:
    process_app_id:
    process_app_name:
    process_app_short_name:
    updated:
}
```

An entry has the following attributes:

- **category** - optional, the category to which the offering belongs.
- **created** - the creation time of the self-service offering, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.
- **description** - optional, a short description of the offering.
- **human_service** - optional, the URL of a IBM Business Process Manager human service (coach), a user interface to provide user input.
- **human_service_app_id** - optional, depending on the human_service attribute, the ID of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_name** - optional, depends on if the human_service attribute is set, the name of the IBM Business Process Manager application to which the human service belongs.
- **human_service_app_short_name** - the short name of the IBM Business Process Manager human service application.
- **icon** - optional, an offering can have an icon assigned that is displayed inside the Self-Service Catalog.
- **id** - the ID of the offering.
- **implementation_type** - the two possible values are 'ibm_bpm_process' and 'script'.
- **name** - the name of the offering.
- **operation_type** - the only possible value is 'service'.
- **ownerid** - the owner of the user who triggered the offering.
- **process** - the name of the IBM Business Process Manager process bound to the offering.
- **process_app_id** - the ID of the IBM Business Process Manager application in which the process is defined.
- **process_app_name** - the name of the IBM Business Process Manager application in which the process is defined.
- **process_app_short_name** - the short name of the IBM Business Process Manager process application.
- **updated** - the time when the self-service offering was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

```json
{
    "isbuiltin": 0,
    "icon": "Web Category Icon:ge100_webcatalog_24",
    "name": "Bucket",
    "id": 8,
    "description": ""
}
```

**Update a category:**

Use this REST call to update a category.

**Available HTTP method**

<table>
<thead>
<tr>
<th>Table 125. Update a category REST API call</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HTTP method</strong></td>
</tr>
<tr>
<td><strong>URL pattern</strong></td>
</tr>
<tr>
<td><strong>Response</strong></td>
</tr>
</tbody>
</table>

Structure of request body:

```json
{
    "<attribute>": "<attribute_value>"
}
```

Delete a category:

Use this REST call to delete a category.

**Available HTTP method**

<table>
<thead>
<tr>
<th>Table 126. Delete a category REST API call</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HTTP method</strong></td>
</tr>
<tr>
<td><strong>URL pattern</strong></td>
</tr>
<tr>
<td><strong>Response</strong></td>
</tr>
</tbody>
</table>
Task engine REST API:

You can use this set of REST API calls to interact with role parameters of a specific server instance.

Task Engine API V1:

The following section describes JSON Formats and instances for task engine API V1.

JSON Formats

Task Response

```
{
    "updated_iso" : "2014-03-31T13:05:14+0200",
    "description_message" : "The process is complete.",
    "domain" : "Default",
    "created" : 1396263830875,
    "error" : {
        "resourceBundle" : "com.ibm.orchestrator.messages.orchestratormessages",
        "message" : null,
        "messageKey" : "BPM_PROCESS_COMPLETE",
        "args" : [
            "3"
        ]
    },
    "user" : "ksadmin",
    "parm" : {
        ...
    },
    "created_iso" : "2014-03-31T13:03:50+0200",
    "status_localized" : "Completed",
    "error_message" : "CTJCO0002I: Business process instance 3 completed successfully.",
    "status" : "COMPLETED",
    "eventTopic" : "com/ibm/orchestrator/serviceinstance/plan/ibm_bpm_process",
    "delayInSeconds" : 30,
    "project" : "admin",
    "id" : "1003",
    "updated" : 1396263914896,
    "description" : {
        "resourceBundle" : "com.ibm.orchestrator.messages.orchestratormessages",
        "message" : null,
        "messageKey" : "PROCESS_COMPLETE",
        "args" : [
        ]
    }
}
```

Tasks Response

[ Task Response 1,......, Task Response n]

GET: Get all tasks

URL method

/kernel/tasks

Accepts

`/*`
Content-Type
application/JSON

Normal Response Codes
200

Response
Tasks Response

GET: Get the task with a given id

URL method
/kernel/tasks/{id}

Accepts
*/*

Content-Type
application/JSON

Normal Response Codes
200

Response
Task Response

List all currently running and recently completed tasks:

Use this REST API method to list all currently running tasks and the tasks that
were completed not longer than two weeks before.

Available HTTP method

| Table 127. List all currently running and recently completed tasks REST API call |
|-------------------------------------------------|-------------------------------------------------|
| HTTP method | GET |
| URL pattern | https://hostname/kernel/tasks/ |
| Response | List all the currently running and recently completed tasks. |
| [task] |
| Return values | • 200 - OK |
| | • 500 - Internal Server Error |

Structure of the query string:
• tasks - a comma-separated list of task objects.

Get entries for a specific task:

Use this REST call to retrieve information about an active task with an indicated
ID.

Available HTTP method

| Table 128. Get information about a specific task |
|-------------------------------------------------|-------------------------------------------------|
| HTTP method | GET |
| URL pattern | https://hostname/kernel/tasks/{id} |
The parameters of the response:

- **updated_iso** - the last update to the task in ISO8601 format.
- **description_message** - G11N enabled information about the function of the task.
- **message** - gives current status, which is displayed to the end user.
- **created** - the time at which the task was created in java timestamp format.
- **error** - a structured object containing an error message if any exists.
- **parm** - a free-form key-value pair object containing all the use-cases specific parameters. It can contain the following parameter:
  - **plan** - contains the self-service offering that was used to create the task. A plan object is only available for tasks that complete self-service offerings.
- **status_localized** - a G11N enabled status for the task.
- **internal status** - one of the following: NEW, QUEUED, RUNNING, SUSPENDED, FAILING, FAILED, COMPLETING, COMPLETED, CANCELING, CANCELED.
- **eventTopic** - identifies the handler that is used to complete the task.
- **description** - contains the internal representation for the **description_message** message.

The following listing shows a sample response that can be obtained via the REST call:
Workload Deployer REST API

You can use this set of REST API calls to interact with the Workload Deployer component.

Application patterns REST API

You can use the REST API to manage your virtual application patterns.

The following tasks can be completed using the REST API.

List all application patterns

GET /resources/applicationPatterns/

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/">https://localhost/resources/applicationPatterns/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
</tbody>
</table>
Table 129. List all application patterns. (continued)

<table>
<thead>
<tr>
<th>Response code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
[
  {
    "content_type": "application/json",
    "last_modifier": "tester",
    "create_time": "2011-02-24T05:41:34Z",
    "last_modified": "2011-02-24T05:41:34Z",
    "access_rights": {
      "tester": "F"
    },
    "content_md5": "661D31C9F14615539E537E9AA5CB02E9",
    "app_type": "application",
    "app_id": "a-faac12d0-23d7-4f57-b3cb-13ce92d5e07f",
    "app_name": "untitled",
    "creator": "tester"
  },
  ...
]
```

Create an application pattern with specific attributes

POST /resources/applicationPatterns/

Different kinds of attributes can be combined. A unique ID is generated for the application.

Table 130. Create an application pattern with specific attributes details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/">https://localhost/resources/applicationPatterns/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request body</td>
<td>Content-Type - application/json (appmodel json file)</td>
</tr>
<tr>
<td>Content-type</td>
<td>Content-Type - application/zip file (zip file including application model and artifacts files)</td>
</tr>
<tr>
<td>Request body example</td>
<td>JSON Application model</td>
</tr>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response header location</td>
<td><a href="https://localhost/resources/applicationPatterns/a-4e21f6e9-2ca7-4a3a-a5cc-00b4f7b7b08">https://localhost/resources/applicationPatterns/a-4e21f6e9-2ca7-4a3a-a5cc-00b4f7b7b08</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Created</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>412</td>
<td>Invalid parameter supplied, for example, the json file is invalid.</td>
</tr>
<tr>
<td>415</td>
<td>Invalid content type</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>
Create an application pattern from an existing application or template (clone)

POST /resources/applicationPatterns/?source={app_id}&app_name={name}&app_type={app_type}

A unique ID is generated for the application.
- Attribute "source": specify application template or application (required)
- Attribute "app_name": specify application name (required)
- Attribute "app_type": specify application type for target application. The values can be application or template. The default value is application.

Table 131. Create an application pattern from an existing application or template (clone) details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/?source=a-679a68f4-6798-424f-8039-1f682f949f45&amp;app_name=testApp">https://localhost/resources/applicationPatterns/?source=a-679a68f4-6798-424f-8039-1f682f949f45&amp;app_name=testApp</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an application name &quot;testApp&quot; from application with id &quot;a-679a68f4-6798-424f-8039-1f682f949f45&quot;</td>
<td></td>
</tr>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response header location</td>
<td><a href="https://localhost/resources/applicationPatterns/a-fb70796e-1b13-467a-babe-b8b700bd563b">https://localhost/resources/applicationPatterns/a-fb70796e-1b13-467a-babe-b8b700bd563b</a></td>
</tr>
<tr>
<td>Response code</td>
<td>201 Created</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>412</td>
<td>Invalid parameter supplied, for example, the application ID is not found.</td>
</tr>
<tr>
<td>415</td>
<td>Invalid content type</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```json
{
  "content_type": "application/json",
  "last_modifier": "tester",
  "create_time": "2011-02-24T05:41:34Z",
  "last_modified": "2011-02-24T05:41:34Z",
  "access_rights": {
    "tester": "F"
  },
  "content_md5": "EF7142254CD653D987E9A9E8A48C01D3",
  "app_type": "application",
  "app_id": "a-4e21f6e9-2ca7-4a3a-a5cc-00f04f7b7f08",
  "app_name": "test",
  "creator": "tester"
}
```
List the application patterns with various filters

GET /resources/applicationPatterns/?<filter>=<filterString>

- Filter "app_name": Filter application pattern with application name.
- Filter "app_type": Filter application pattern with application type. filterString can be application, template or service (for shared service). If filterString is null or empty, all application pattern are returned.

Important: The result lists only the application with R access for the request.
- Filter “patterntype” and “version”: Filter application patterns with versioned pattern type, for example, ?patterntype=webapp&version=1.0

Table 132. List application pattern with various filters details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/?app_name=web%20application&amp;app_type=template">https://localhost/resources/applicationPatterns/?app_name=web%20application&amp;app_type=template</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve all application templates whose name contains string &quot;web application&quot;.</td>
<td></td>
</tr>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:
```
[
  {
    "content_type": "application/json",
    "last_modifier": "cbadmin",
    "create_time": "2011-02-24T05:41:34Z",
    "last_modified": "2011-02-24T05:41:34Z",
    "access_rights": {
      "AllUsers": "R"
    },
    "content_md5": "2565A57FC1149260B869ADCF7C48154E",
    "app_type": "template",
    "app_id": "a-14057bbd-9651-461a-a597-5df8957110fc",
    "app_name": "web application",
    "creator": "cbadmin"
  },
  ...
]
```

Update application pattern

PUT /resources/applicationPatterns/{app_id}

Table 133. Update application pattern details.

| Example URL | https://localhost/resources/applicationPatterns/a-cdaac959-672c-4d7f-a648-b333a3843422 |
Table 133. Update application pattern details. (continued)

<table>
<thead>
<tr>
<th>Response content-type and body</th>
<th>Content-Type - application/json (body is the application model json file)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Content-Type - application/zip (body is zip file, including the application model and artifacts files)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Response content-type</th>
<th>application/json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>401 The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403 Access forbidden</td>
</tr>
<tr>
<td></td>
<td>404 The application specified by {appID} is not found.</td>
</tr>
<tr>
<td></td>
<td>412 Invalid parameter supplied, for example, the json file is invalid.</td>
</tr>
<tr>
<td></td>
<td>500 Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
{
  "content_type": "application/json",
  "last_modifier": "tester",
  "create_time": "2011-02-24T05:41:34Z",
  "last_modified": "2011-02-24T05:41:34Z",
  "access_rights": {
    "tester": "F"
  },
  "content_md5": "5B8F7E6CF56F7CE804788C0086589AFF",
  "app_type": "application",
  "app_id": "a-fb70796e-1b13-467a-babe-b8b700bd563b",
  "name": "App for Testing",
  "locked": "false",
  "creator": "tester"
}
```

**Return detailed information about the application pattern**

GET /resources/applicationPatterns/{app_id}

Table 134. Return detailed information about the application pattern.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/a-679a68f4-6798-424f-8039-1f682f949f45">https://localhost/resources/applicationPatterns/a-679a68f4-6798-424f-8039-1f682f949f45</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>401 The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403 Access forbidden</td>
</tr>
<tr>
<td></td>
<td>404 The application specified by {appID} is not found.</td>
</tr>
<tr>
<td></td>
<td>500 Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
Download the application pattern zip file, including all artifacts and the json file

GET /resources/applicationPatterns/{app_id}?zip

Table 135. Download the application pattern zip file, including all artifacts and the json file details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/a-679a68f4-6798-424f-8039-1f682f949f45?zip">https://localhost/resources/applicationPatterns/a-679a68f4-6798-424f-8039-1f682f949f45?zip</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response body</td>
<td>Application zip file</td>
</tr>
<tr>
<td>Response code</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
<tr>
<td></td>
<td>403: Access forbidden</td>
</tr>
</tbody>
</table>

Update application pattern access right for the specified user name or group name

PUT /resources/applicationPatterns/{app_id}/accessRights/{name}?{ user or group }

Table 136. Update application pattern access right details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/a-cdaac959-672c-4df7-a648-b333a3843422/accessRights/Everyone?group">https://localhost/resources/applicationPatterns/a-cdaac959-672c-4df7-a648-b333a3843422/accessRights/Everyone?group</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Access forbidden</td>
</tr>
<tr>
<td></td>
<td>403: Access forbidden</td>
</tr>
</tbody>
</table>
Table 136. Update application pattern access right details. (continued)

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>412</td>
<td>Invalid parameter supplied, for example, the json file is invalid.</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
{ "access_rights": "F" }
```

Delete a specified application pattern

DELETE /resources/applicationPatterns/{app_id}

Response example:

```
https://localhost/resources/applicationPatterns/a-cdaac959-672c-4df7-a648-b333a3843422
```

Table 137. Delete a specified application pattern details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th>Response content-type</th>
<th>Response code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>application/json</td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td></td>
<td></td>
<td>409</td>
<td>Conflict</td>
</tr>
<tr>
<td></td>
<td></td>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Artifacts REST API

List all artifacts of a given application pattern

GET /resources/applicationPatterns/{app_id}/artifacts/

Response example:

```
[
  {
    "content_type": "application/octet-stream",
    "last_modifier": "tester",
```

Table 138. List all artifact of a given application pattern details.
Upload an artifact file
PUT /resources/applicationPatterns/{app_id}/artifacts/{filename}

The file is overwritten if the file already exists.

Table 139. Upload an artifact file details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/792f2265-2a1d-411a-b5d6-c40d658539a1/artifacts/test.war">https://localhost/resources/applicationPatterns/792f2265-2a1d-411a-b5d6-c40d658539a1/artifacts/test.war</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request content-type</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>Request body</td>
<td>Artifact file</td>
</tr>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Application specified by {appID} is not found.</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:
{
   "file": "artifacts/test.war"
}

Download an artifact file
GET /resources/applicationPatterns/{app_id}/artifacts/{filename}?download

Table 140. Download an artifact file details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/792f2265-2a1d-411a-b5d6-c40d658539a1/artifacts/test.war?download">https://localhost/resources/applicationPatterns/792f2265-2a1d-411a-b5d6-c40d658539a1/artifacts/test.war?download</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/octet-stream</td>
</tr>
<tr>
<td>Response body</td>
<td>Artifact file</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>404</td>
<td>Application specified by {appID} is not found.</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>
Get the detail of the artifact

GET /resources/applicationPatterns/{app_id}/artifacts/{filename}

Table 141. Get the detail of the artifact.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/792f2265-2a1d-411a-b5d6-c40d658539a1/artifacts/test.ddl">https://localhost/resources/applicationPatterns/792f2265-2a1d-411a-b5d6-c40d658539a1/artifacts/test.ddl</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>401</td>
</tr>
<tr>
<td></td>
<td>403</td>
</tr>
<tr>
<td></td>
<td>404</td>
</tr>
<tr>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>

Response example:

```json
{
    "content_type": "application/octet-stream",
    "last_modifier": "cbadmin",
    "create_time": "2011-05-14T06:04:36Z",
    "last_modified": "2011-05-14T06:04:36Z",
    "access_rights": {
        "cbadmin": "F",
        "_group_:Everyone": "R"
    },
    "content_md5": "68C84965B52D8BC66D5DCB7CD9E2B774",
    "creator": "cbadmin"
}
```

Delete an artifact file

DELETE /resources/applicationPatterns/{app_id}/artifacts/{filename}

Table 142. Delete an artifact file details.

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/applicationPatterns/792f2265-2a1d-411a-b5d6-c40d658539a1/artifacts/test.war">https://localhost/resources/applicationPatterns/792f2265-2a1d-411a-b5d6-c40d658539a1/artifacts/test.war</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>401</td>
</tr>
<tr>
<td></td>
<td>403</td>
</tr>
<tr>
<td></td>
<td>404</td>
</tr>
<tr>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>

Note: If the application specified by {app_id} or the artifact specified by {filename} is not found, a 200 response code returns, response body: {"success": "false"}
Environment profiles REST API

You can use the representational state transfer (REST) application programming interface (API) to manage environment profiles.

Available HTTP Methods

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Date Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/resources/environmentProfiles</td>
<td>application/json</td>
<td>200</td>
<td>403 This code is returned if the requester does not have access to list environment profiles. 500 This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
<tr>
<td>POST</td>
<td>/resources/environmentProfiles</td>
<td>application/json</td>
<td>201</td>
<td>500 This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
</tbody>
</table>

An environment profile has the following attributes:

created
Specifies the creation time of the environment profile, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

currentmessage
Specifies the message associated with the current status of the environment profile. This is an 8 character string value that is generated by the product.

currentmessage_text
Specifies the textual representation of currentmessage. This is a string representation of currentmessage in the preferred language of the requester and is automatically generated by the product.

currentstatus
Specifies a string constant representing the current status of the environment profile. This is an 8 character string value is automatically generated by the product.

currentstatus_text
Specifies the textual representation of currentstatus. This is a string representation of currentstatus in the preferred language of the requester and is automatically generated by the product.
description
Specifies the description of the environment profile. This field is a string value with a maximum of 1024 characters.

domainname
Specifies the name of the domain.

domainou
Specifies the organizational unit where the computer account is stored.

domainpassword
Specifies the password of the domain user.

domainusername
Specifies the user name that is authorized to add a computer account the domain.

environment
Specifies the environment the profile represents. Valid values from 0 to 7 are:
- deployer.environmentprofile.ALL_ENVIRONMENT
- deployer.environmentprofile.DEVELOPMENT_ENVIRONMENT
- deployer.environmentprofile.TEST_ENVIRONMENT
- deployer.environmentprofile.QUALITY_ASSURANCE_ENVIRONMENT
- deployer.environmentprofile.PERFORMANCE_ENVIRONMENT
- deployer.environmentprofile.RESEARCH_ENVIRONMENT
- deployer.environmentprofile.PRODUCTION_ENVIRONMENT
- deployer.environmentprofile.PRE_PRODUCTION_ENVIRONMENT

environment_text
Specifies the textual representation of environment. This is a string representation of environment in the preferred language of the requester and is automatically generated by the product.

id
Specifies the ID of the environment profile. This numeric value is automatically generated by the product.

kmsipaddress
Specifies the IP address of the KMS server in your environment.

kmsport
Specifies the port used for KMS service.

name
Specifies the display name associated with this environment profile. This field contains a string value with a maximum of 1024 characters.

owner
Specifies the uniform resource identifier (URI) of the user that owns this environment profile. The URI is relative and should be resolved against the URI of the owner.

platform
Specifies the type of hypervisors this environment profile supports on deployments. Valid values are ESX, PowerVM, and zVM.

updated
Specifies the time the environment profile was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

vmname_pattern
Specifies the pattern used to generate virtual machine names.
GET /resources/environmentProfiles example

[
  {
    "clouds": [
      {
        "cloud": "/resources/clouds/1",
        "alias": "t",
        "ipGroups": [
          {
            "alias": "t",
            "ipGroup": "/resources/ipGroups/1"
          }
        ]
      }
    ],
    "created": 1285348986268,
    "currentMessage": "RM25008",
    "currentMessageText": "Environment profile can now be used for deployments"
  }
],

A virtual system instance has the following attributes:

- "currentStatus": "RM01001",
- "currentStatusText": "Defined",
- "description": "",
- "environment": 0,
- "environmentText": "All",
- "platform": "OpenStack",
- "updated": 1285352832052,
See the description of GET /resources/virtualsystems/{id} for attribute details.

**GET /resources/environmentProfiles/{id} example**

```json
{
   "clouds": [
      {
         "cloud": "/resources/clouds/1",
         "alias": "t",
         "ipGroups": [
            {
               "alias": "t",
               "ipGroup": "/resources/ipGroups/1"
            }
         ]
      },
      {
         "cloud": "/resources/clouds/1",
         "alias": "t",
         "ipGroups": [
            {
               "alias": "t",
               "ipGroup": "/resources/ipGroups/1"
            }
         ]
      }
   ],
   "created": 1285348986268,
   "currentmessage": "RM25008",
   "currentmessage_text": "Environment profile can now be use for deployments"
   "currentstatus": "RM01001",
   "currentstatus_text": "Defined",
   "description": "",
   "environment": 0,
   "environment_text": "All",
   "id": 2,
   "ipsource": 0,
   "ipsource_text": "Workload Deployer",
   "name": "test2",
   "owner": "/resources/users/1",
   "platform": "OpenStack",
   "updated": 1285352832052,
   "vmname_pattern": "",
}
```

**POST /resources/environmentProfiles example**

Request JSON:

```json
{
   "platform": "OpenStack",
   "environment": 0,
   "description": "Sample description",
   "name": "envProfileName"
}
```

**Log viewer manager REST API**

You can use the representational state transfer (REST) application programming interface (API) to manage Log Viewer Manager.
### Available HTTP Methods

**Table 144. REST API for LogViewerMgr**

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/resources/logViewerMgr</td>
<td></td>
<td>200</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This code is returned if log viewing for the specified log file was successfully initialized. The Location header in the response contains a URL that can be queried to view contents of the log file.</td>
<td>This code is returned if the requester has not been assigned the admin role.</td>
</tr>
<tr>
<td>GET</td>
<td>/resources/logViewerMgr/{id}</td>
<td>application/json</td>
<td>200</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This code is returned if content from the log file is included in the output.</td>
<td>This code is returned if the requester has not been assigned the admin role.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>204</td>
<td>404</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This code is returned if the specified startPoint and lineCount do not include any content from the log file.</td>
<td>This code is returned if the specified log viewing has not been initialized correctly.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This code is returned if the IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
</tbody>
</table>

**POST /resources/logViewerMgr example**

This REST API call initializes log viewing for a specific log file. The Location header in the response contains a URL that can be subsequently queried to fetch new contents of the specified file.

Request:
POST /resources/logViewerMgr?logfile=trace/trace.log

Response headers:
Location: https://myproduct.mycompany.com/resources/logViewerMgr/trace_%5E_trace.log

**GET /resources/logViewerMgr/{id} example**

Request:
GET /resources/logViewerMgr/trace_%5E_trace.log?startingPoint=0&lineCount=3
The TAIL_CONTENT entry in the response contains contents of the log file; the NEXT_CHUNK value in the response can be used as the startingPoint in the next request to retrieve subsequent content from the log file.

Related tasks:

“REST API reference” on page 655

The representational state transfer (REST) application programming interface (API) is provided by IBM Cloud Orchestrator.

Logging REST API

Use REST APIs to log application monitoring results.

The following tasks can be completed using the REST API.

List all the logs on a specific virtual machine

GET /resources/virtualApplications/\{virtual_application_instance_id\}/logs/virtualMachines/\{virtual_machine_id\}

<table>
<thead>
<tr>
<th>Table 145. List all the logs on a specific virtual machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URL</td>
</tr>
<tr>
<td>Response content-type</td>
</tr>
<tr>
<td>Response code</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Response example (lists of files for different roles):

```json
{
  "OS" : ["/var/log/brcm-iscsi.log",
           "/var/log/cron",
           "/var/log/messages",
           "/var/log/secure",
           "/var/log/acpid",
           "/var/log/dmesg",
           "/var/log/yum.log",
           "/var/log/wtmp",
           "/var/log/spooler",
           "/var/log/maillog",
           "/var/log/audit.log",
           "/var/log/secure.log",
           "/var/log/kern.log", 
           "/var/log/djmage.log", 
           "/var/log/j景德e.log",
           "/var/log/audit.log",
           "/var/log/secure.log", 
           "/var/log/kern.log",
           "/var/log/djmage.log", 
           "/var/log/j景德e.log"
  ],
  "WAS" : ["/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/SystemErr.log",
            "/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/native_stderr.log",
            "/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/native_stdout.log",
            "/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/SystemOut.log",
            "/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/ffdc/ffdc.3819090375058668621.txt",
            "/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/ffdc/FfdcSummary.txt",
            "/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/ffdc/ffdc.6804038344002622019.txt"
  ],
  "IWD Agent" : ["/opt/IBM/maestro/agent/usr/servers/Web_Application-was.11319468974926/logs/Web_Application-was.11319468974926.MONITORING/trace.log"
```
Get the content of a specific log file

GET /resources/virtualApplications/{virtual_application_instance_id}/logs/virtualMachines/{virtual_machine_id} /{log_absolute_path}

Table 146. Get the content of a specific log file

<table>
<thead>
<tr>
<th>Example URL</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/logs/virtualMachines/Web_Application-was.11319468974926/opt/IBM/maestro/agent/usr/servers/Web_Application-was.11319468974926/logs/Web_Application-was.11319468974926.WAS/trace.log">https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/logs/virtualMachines/Web_Application-was.11319468974926/opt/IBM/maestro/agent/usr/servers/Web_Application-was.11319468974926/logs/Web_Application-was.11319468974926.WAS/trace.log</a></td>
</tr>
</tbody>
</table>

| Request headers | bytes=[start]-[end] | For example, bytes=0-500. Specify the byte range of the log file to get. If the byte range is not set, entire log file is gotten. |

<table>
<thead>
<tr>
<th>Response code</th>
<th>200</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
<td></td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
<td></td>
</tr>
</tbody>
</table>

Response example:

[2011-10-24 16:03:53,756] WAS/start.py 47377699650752 pid=23164 INFO WAS: 8.0.0.1
[2011-10-24 16:05:53,998] WAS/start.py 47377699650752 pid=23164 INFO set WAS role status to RUNNING
Monitoring REST API
Use REST APIs for applications monitoring tasks.

The following tasks can be completed using the REST API:

**Get the overall monitoring status for a given virtual application instance**

GET /resources/virtualApplications/{virtual_application_instance_id}/monitoring

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring">https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```json
{
  "SERVERS": [
    {
      "vm_id": "1",
      "vm_uuid": "4217cc6c-0d82-575d-c983-4c76d221de65",
      "hypervisor_uuid": "52e11db5-2c40-e011-ae1b-00215e5d6968",
      "private_ip": "172.16.71.208",
      "server_name": "Web_Application-was.11319468974926",
      "public_hostname": "ipas-vm-071-208.purescale.raleigh.ibm.com",
      "state": "RUNNING",
      "time_stamp": "1319471278884",
      "vm_name": "Web_Application-was.11319468974926",
      "deployment_id": "d-65d29715-9063-436d-8593-d5218208f8aa",
      "hypervisor_hostname": "172.16.64.31",
      "availability": "NORMAL",
      "public_ip": "172.16.71.208"
    }
  ],
  "application": {
    "connectors": [],
    "workload": "TRUE",
    "application_name": "simple",
    "application_id": "a-0f905e-d5f2-4512-b9ae-e4934a8e3ea0"
  },
  "ROLETYPES": [
    {
      "roleType": "AGENT",
      "template": "Web_Application-was",
      "availability": "NORMAL"
    },
    {
      "roleType": "SSH",
      "template": "Web_Application-was",
      "availability": "NORMAL"
    },
    {
      "roleType": "MONITORING",
      "template": "Web_Application-was",
      "availability": "NORMAL"
    },
    {
      "roleType": "WAS",
```
Get the virtual machine monitoring status for a given virtual application instance

GET /resources/virtualApplications/\{virtual_application_instance_id\}/monitoring/servers

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring/servers">https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring/servers</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200, 401, 403, 500</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
[
  {
    "vm_id":"1",
    "vm_uuid":"4217cc6c-0d82-575d-c983-4c76d21ded5",
    "hypervisor_uuid":"52e11db5-2c40-e011-aeb1-00215e5d6968",
    "private_ip":"172.16.71.208",
    "server_name":"Web_Application-was.11319468974926",
    "pattern_version":2.0,
    "pattern_type": "webapp",
    "availability": "NORMAL"
  }
]```
Get the middleware monitoring status for a given virtual application instance

GET /resources/virtualApplications/{virtual_application_instance_id}/monitoring/roles

Table 149. Get the middleware monitoring status for a given virtual application instance

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring/role">https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring/role</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```json
[
  {
    "time_stamp":1319472359280,
    "state":"RUNNING",
    "private_ip":"172.16.71.208",
    "role_type":"WAS",
    "role_name":"Web_Application-was.11319468974926.WAS",
    "display_metrics":true,
    "server_name":"Web_Application-was.11319468974926",
    "pattern_version":"2.0",
    "pattern_type":"webapp",
    "availability":"NORMAL"
  }
]
```

Get virtual machine level monitoring metrics for a specific virtual machine

GET /resources/virtualApplications/{virtual_application_instance_id}/monitoring/servers/{virtual_machine_id}/metrics/

Table 150. Get virtual machine level monitoring metrics of a specific virtual machine

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring/servers/Web_Application-was.11319468974926/metrics/">https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring/servers/Web_Application-was.11319468974926/metrics/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:
Get middleware level monitoring metrics of a specific middleware

GET /resources/virtualApplications/{virtual_application_instance_id}/monitoring/roles/{middleware_id}/metrics/

Table 151. Get middleware level monitoring metrics of a specific middleware

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring/roles/Web_Application-was.11319468974926.WAS/metrics/">https://localhost/resources/virtualApplications/d-65d29715-9063-436d-8593-d5218208f8aa/monitoring/roles/Web_Application-was.11319468974926.WAS/metrics/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
{
  "WAS_WebApplications":{
    "time_stamp":1319538432348,
    "max_service_time":0,
    "min_service_time":0,
    "service_time":0,
    "request_count":0
  },
  "WAS_TransactionManager":{
    "time_stamp":1319538432348,
    "rollbacked_count":0,
    "active_count":0,
    "committed_count":12
  },
  "WAS_JDBCConnectionPools":{
    "time_stamp":1319538432348,
    "max_percent_used":0,
    "min_percent_used":0,
    "percent_used":0,
    "wait_time":0,
    "min_wait_time":0,
    "max_wait_time":0
  },
  "WAS_JVMRuntime":{
```
Pattern types REST API

You can use the REST API to manage your virtual application pattern types.

The following tasks can be completed using the REST API.

**List all pattern types with version format "vr" or "vmf"**

GET /resources/patternTypes/?version={format}

"vr" is to get pattern types with "vr" version format, e.g. "1.0"

"vmf" is to get pattern types with "vmf" version format, e.g. "1.0.0.0"

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/patternTypes/?version=vr">https://localhost/resources/patternTypes/?version=vr</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
[
  {
    "license": {
      "type": "PVU",
      "pid": "5725E00"
    },
    "status": "avail",
    "licenses": [
      "https://172.16.33.45:9444/storehouse/admin/patterntypes/dbaas/1.0/licenses/"
    ],
    "shortname": "dbaas",
    "version": "1.0",
    "name": "DBaaS Pattern Type",
    "description": "IBM Workload Deployer Pattern Type for DBaaS",
    "url": "https://172.16.33.45:9444/storehouse/admin/patterntypes/dbaas/1.0/"
  },
  ...
]
```

**Create a pattern type**

POST /resources/patternTypes

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/patternTypes/">https://localhost/resources/patternTypes/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Request example</td>
<td>Request body: tgz File</td>
</tr>
<tr>
<td>Response header location</td>
<td><a href="https://localhost/resources/patternTypes/">https://localhost/resources/patternTypes/</a> [patternTypesName]/[version_vrmf]</td>
</tr>
<tr>
<td>Response code</td>
<td>201 Created successfully</td>
</tr>
</tbody>
</table>
Table 153. Create a pattern type details (continued)

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

List detail information of one pattern type

GET /resources/patternTypes/{patternTypeName}/{version}

Table 154. List detail information of one pattern type

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/patternTypes/dbaas/1.0">https://localhost/resources/patternTypes/dbaas/1.0</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>401 The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403 Access forbidden</td>
</tr>
<tr>
<td></td>
<td>500 Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```json
{
  "license": {
    "type": "PVU",
    "pid": "5725E00"
  },
  "status": "avail",
  "licenses": [
    "https://172.16.33.45:9444/storehouse/admin/patterntypes/dbaas/1.0/licenses/
  ],
  "shortname": "dbaas",
  "version": "1.0",
  "name": "DBaaS Pattern Type",
  "description": "IBM Workload Deployer Pattern Type for DBaaS",
  "url": "https://172.16.33.45:9444/storehouse/admin/patterntypes/dbaas/1.0/"
}
```

List plugin list of one pattern type

GET /resources/patternTypes/{patternTypeName}/{version}/plugins

Table 155. List plugin list of one pattern type details

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/patternTypes/dbaas/1.0/plugins">https://localhost/resources/patternTypes/dbaas/1.0/plugins</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>401 The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403 Access forbidden</td>
</tr>
<tr>
<td></td>
<td>500 Unexpected error</td>
</tr>
</tbody>
</table>

Response example:
Accept the license agreement of a pattern type

PUT /resources/patternTypes/{patternTypeName}/{version_vr}/

Table 156. Accept the license agreement of a pattern type

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/patternTypes/webapp/1.0/">https://localhost/resources/patternTypes/webapp/1.0/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
</tbody>
</table>
| Request example | Request body:
  `{  
  "status": "accepted"
  }`  
  Valid status includes "accepted", "avail", "unavail" |
| Response body | `{  
  "status": "accepted"
  }` |
| Response code | 200 | OK |
|              | 401 | The user is not authorized to perform this action. |
|              | 403 | Access forbidden |
|              | 500 | Unexpected error |

Delete a pattern type

DELETE /resources/patternTypes/{patternTypeName}/{version_vr}|

Table 157. Delete a pattern type details

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/patternTypes/webapp/1.0.0">https://localhost/resources/patternTypes/webapp/1.0.0</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response example</td>
<td>true or false</td>
</tr>
<tr>
<td>Response code</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>401</td>
</tr>
<tr>
<td></td>
<td>403</td>
</tr>
<tr>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>
Patterns REST API
You can use the representational state transfer (REST) application programming interface (API) to manage patterns.

Available HTTP Methods

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Date Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/resources/patterns</td>
<td>application/json</td>
<td>200</td>
<td>Returns the list of patterns that are visible to the client. If a ‘name=’ query parameter was supplied, the list of patterns will contain only patterns whose name contains the specified search string.</td>
</tr>
<tr>
<td>GET</td>
<td>/resources/patterns/[id]</td>
<td>application/json</td>
<td>200</td>
<td>Returns the pattern associated with the given ID.</td>
</tr>
</tbody>
</table>

A pattern has the following attributes:

created
Specifies the creation time of the pattern, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

currentmessage
Specifies the message associated with the current status of the pattern. This field is an 8 character string value that is automatically generated by the product.

currentmessage_text
Specifies the textual representation of currentmessage. This is a string representation of currentmessage in the preferred language of the requester and is automatically generated by the product.
currentstatus
Specifies a string constant representing the current status of the pattern. This field is an 8 character string value that is automatically generated by the product.

currentstatus_text
Specifies the textual representation of currentstatus. This is a string representation of currentstatus in the preferred language of the requester and is automatically generated by the product.

description
Specifies the description of the pattern. This field is a string value with a maximum of 1024 characters.

id
Specifies the ID of the pattern. This numeric value is automatically generated by the product.

name
Specifies the name of the pattern. This field is a string value with a maximum of 1024 characters.

owner
Specifies the uniform resource identifier (URI) of the user that owns this pattern. The URI is relative and should be resolved against the URI of the pattern.

parts
Specifies a list containing one map per part contained in the pattern.
The map for each part contains the following attributes:

count
The number of virtual machines that are created from this part when the pattern is deployed. A value of null indicates that the part can only be used to construct a single virtual machine. Parts that can be used to construct multiple virtual machines will have a positive integer value for this attribute.

description
Specifies a textual description of the part.

id
Specifies the ID of the pattern part. This numeric value is automatically generated by the product.

label
Specifies displayable text used to identify the part.

properties
Specifies a list containing one map per property defined for the part.

scripts
Specifies a list containing one map per script defined for the part.

virtualimage
Specifies the uniform resource identifier (URI) of the virtual image associated with the part. The URI is relative and should be resolved against the URI of the pattern.
The map for each part property contains the following attributes:

description
Specifies a textual description of the property.

key
Specifies a string that uniquely identifies the property within the part property.

label
Specifies displayable text used to identify the property.

class
Specifies a string value used to identify related properties within a
part. The combination of pclass and key values is unique for every property contained in a given part.

**type**
Specifies a string indicating the type of values that may be assigned to this property. The value will be one of "string", "integer" or "boolean".

**validValues**
For properties that are only allowed to have certain values, the `validValues` attribute contains a list of the allowable values.

**value**
Specifies the default value for the property. The type of this value depends on the property's type.

The map for each script contains the following attributes:

**description**
Specifies a textual description of the script.

**id**
Specifies the ID of the pattern script. This numeric value is automatically generated by the product.

**label**
Specifies the displayable text used to identify the script.

**parameters**
Specifies a list containing one map per parameter defined for the script.

The map for each parameter contains the following attributes:

**key**
Specifies a string that uniquely identifies the parameter. The script key is a string with a maximum length of 4098 characters.

**value**
The default value for the parameter. All parameter have string values with a maximum length of 4098 characters.

**updated**
Specifies the time the pattern was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

**virtualsystems**
Specifies the list of URIs of the virtual system instances using this pattern. The URIs are relative should be resolved against the URI of the pattern that contains them.

**GET /resources/patterns example**

```json
[
  {
    "created": 1369136077948,
    "version": "1.0.0",
    "currentstatus": "RM01027",
    "updated": 1369136077948,
    "name": "WASPattern",
    "id": 2,
    "description": null,
    "currentmessage": null,
    "ownerid": 2
  },
  {
    "created": 1369383745603,
    "version": "1.0.0",
    "currentstatus": "RM01027",
    "updated": 1369383745603,
    "name": "ConnectedImages",
    "id": 6,
    "description": null,
```
See the description of `GET /resources/patterns/{id}` for attribute details.

**GET /resources/patterns/{id} example**

```
{
  "path": null,
  "created": 1369136077975,
  "name": "WASPattern",
  "currentstatus": "RM01027",
  "validationstatus": "RM01001",
  "ownerid": 2,
  "currentmessage": null,
  "validationmessage": "RM06000",
  "constraints": [
  
  ],
  "validation": [
    {
      "status": "RM01001",
      "message": "RM06000"
    }
  ],
  "id": 2,
  "updated": 1369145895473,
  "counter": 0,
  "description": null
}
```

**Note:** Key-value pairs that are only used by user interface clients are optional.

**GET /resources/patterns/{id}?parts=true example**

```
{
  "parts": [
    {
      "description": "The IBM HTTP server as the Web server",
      "memberCount": 1,
      "key": "IHSPart",
      "partType": "IHS Only Node",
      "partVersion": "1.0.0",
      "partCaption": "IBM HTTP servers",
      "label": "IBM HTTP servers",
      "id": 1,
      "resid": 6,
      "partDescriptiveName": null,
      "elasticDisabled": false,
      "elastic": true,
      "renderColor":="#F2F9E4",
      "renderBackgroundImage": "images.zip/bg_f2f9e4.png",
      "renderConfigScriptColor":="#E4EFCC",
      "renderIcon": "images.zip/http-server-icon.gif",
      "renderListIcon": "images.zip/http-server-icon-large.gif",
      "requiresAdvancedFunctionEnabled": false,
      "supportsIPv6": true,
      "hypervisorType": "VMware-ESXi",
      "properties": [
        {
          "description": "Hypervisor for which image is created",
          "label": "Hypervisor",
          "id": "1",
          "key": "Hypervisor",
          "value": "VMware-ESXi",
```
The representational state transfer (REST) application programming interface (API) is provided by IBM Cloud Orchestrator.

**Plug-ins REST API**
You can use the REST API to manage your plug-ins.

The following tasks can be completed using the REST API.

### Retrieve all plug-ins

**GET /resources/plugins/**

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/plugins/">https://localhost/resources/plugins/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>Response content-type</td>
<td>OK</td>
</tr>
<tr>
<td>Response code</td>
<td>401</td>
</tr>
<tr>
<td>Response content-type</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>Response code</td>
<td>403</td>
</tr>
<tr>
<td>Response content-type</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>Response code</td>
<td>500</td>
</tr>
<tr>
<td>Response content-type</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
[  {  
      "content_type": "application/json",
      "last_modified": "cbadmin",
      "create_time": "2011-02-23T13:35:55Z",
      "enabled": true,
      "last_modified": "2011-02-23T13:38:48Z",
      "access_rights": {  
          "cbadmin": "F",
          "_group_:Everyone": "R"
      },
      "content_md5": "6DDE51DF49D718372BA1EBAFF3E71410",
      "name": "waswmqq/1.0.0.0",
      "creator": "cbadmin"
  },
  {  
      "content_type": "application/json",
      "last_modified": "cbadmin",
      "create_time": "2011-02-23T13:36:08Z",
      "enabled": true,
      "last_modified": "2011-02-23T13:36:09Z",
      "access_rights": {  
          "cbadmin": "F",
          "_group_:Everyone": "R"
      },
      "content_md5": "6DDE51DF49D718372BA1EBAFF3E71410",
      "name": "waswmqq/1.0.0.0",
      "creator": "cbadmin"
  }
]  ```
Create a plug-in

POST /resources/plugins/

Table 160. Create a plug-in details

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/plugins/">https://localhost/resources/plugins/</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Request content-type</td>
<td>application/binary</td>
</tr>
<tr>
<td>Request example</td>
<td>Request body: the tgz file, for example, firewall-1.0.0.0.tgz</td>
</tr>
<tr>
<td>Response header location</td>
<td><a href="https://localhost/resources/plugins/firewall/1.0.0.0">https://localhost/resources/plugins/firewall/1.0.0.0</a></td>
</tr>
<tr>
<td>Response code</td>
<td>201 Created successfully</td>
</tr>
<tr>
<td></td>
<td>401 The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403 Access forbidden</td>
</tr>
<tr>
<td></td>
<td>409 Conflict</td>
</tr>
<tr>
<td></td>
<td>500 Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
{
   "artifacts": "https://172.16.33.84:9444/storehouse/admin/plugins/firewall/1.0.0.0/",
   "enabled": true,
   "plugin": "https://172.16.33.84:9443/services/plugins/firewall/1.0.0.0",
   "ETag": "\"177436A9E6C767F309C20D8158F8587-2011-05-14T14:13:25Z-1305382405351\"",
   "patternTypes": null,
   "name": "firewall/1.0.0.0"
}
```

Delete a plug-in

DELETE /resources/plugins/{plugin_name}/{version}

Table 161. Delete a plug-in details

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/plugins/firewall/1.0.0.0">https://localhost/resources/plugins/firewall/1.0.0.0</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td></td>
<td>401 The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403 Access forbidden</td>
</tr>
<tr>
<td></td>
<td>409 Conflict</td>
</tr>
<tr>
<td></td>
<td>500 Unexpected error</td>
</tr>
</tbody>
</table>
Retrieve plug-in information
GET /resources/plugins/{plugin_name}/{version}

Table 162. Retrieve plug-in information details

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/plugins/firewall/1.0.0.0">https://localhost/resources/plugins/firewall/1.0.0.0</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>404</td>
<td>The Plug-in specified by the [plugin_name] is not found.</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:
{
  "Content-MD5": "177436A9E6C767F396C2D1D8158F8587",
  "name": "firewall/1.0.0.0",
  "AccessRights": {
    "cbadmin": "F",
    "_group_:Everyone": "R"  
  },
  "Creator": "cbadmin",
  "LastModifier": "cbadmin",
  "Content-Type": "application/json",
  "LastModified": "2011-05-14T14:13:25Z"
}

Shared services REST API
Use REST APIs to complete tasks related to the shared services pattern.

The following tasks can be completed using the REST API.

List all patterns of shared services
GET /resources/sharedServices/

Table 163. List all patterns of shared services

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/sharedServices">https://localhost/resources/sharedServices</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:
[
  {
    "last_modifier": "cbadmin",
    "content_type": "application/json",
    "service_version": "1.0",
    "app_storehouse_base_url": "https://172.16.65.130:9444/storehouse/user/applications/a-098b"  
}]

Chapter 10. Reference  765
Deploy shared services into a cloud group

Important: After a shared services pattern is deployed, an instance is generated of shared services pattern. The instance is a virtual application instance, therefore you can use the virtual application instance REST API to get the details.

POST /resources/sharedServices/<id>/virtualApplications

Table 164. Deploy a shared services pattern into a cloud group

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/sharedServices/a-6d29ddbc-7005-469a-878f-b467ff57dd3f/virtualApplications">https://localhost/resources/sharedServices/a-6d29ddbc-7005-469a-878f-b467ff57dd3f/virtualApplications</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>400 Invalid request parameter.</td>
</tr>
<tr>
<td></td>
<td>401 The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>409 Instance exists in the cloud.</td>
</tr>
<tr>
<td></td>
<td>500 Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
{
  "deployment_name":"PROXY",
  "ssh_keys":null,
  "model":{
    "model":{
      "servicename":"proxy",
      "nodes":[
        {
          "attributes":{
            "numberOfELBInstances":2
          },
          "type":"PROXY",
          "id":"sharedservice",
          "groups":null
        }
      ]
    }]
  }
}```
Version REST API
You can use the representational state transfer (REST) application programming interface (API) to obtain version information about IBM Cloud Orchestrator.

Available HTTP Methods

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/resources/version</td>
<td>application/json</td>
<td>200</td>
<td>None.</td>
</tr>
</tbody>
</table>

The version information is returned as a map containing the following keys:

version
The version of the Workload Deployer component, as a 4-part version number followed by a hyphen and a build identifier.

GET /resources/version example
{
"version": "3.0.0.0-32058"
}

Related tasks:
“REST API reference” on page 655
The representational state transfer (REST) application programming interface (API) is provided by IBM Cloud Orchestrator.
Virtual appliance instances REST API

You can use the representational state transfer (REST) application programming interface (API) to manage virtual appliance instances. These images are the virtual machines that are not created by IBM Cloud Orchestrator but that were taken over.

Available HTTP Methods

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/resources/virtualApplianceInstances</td>
<td>application/json</td>
<td>200</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>/resources/virtualApplianceInstances/{id}</td>
<td>application/json</td>
<td>200</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

403 This code is returned if the requester does not have access to list virtual appliance instances.

500 This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.
<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT</td>
<td>/resources/virtualApplianceInstances/{id}</td>
<td>application/json</td>
<td>200</td>
<td>400</td>
<td>This code is returned if there are problems parsing the JSON data in the request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>403</td>
<td>This code is returned if the requester does not have permission to update the virtual appliance instance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>404</td>
<td>This code is returned if the request references a resource that is not defined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
<tr>
<td>DELETE</td>
<td>/resources/virtualApplianceInstances/{id}</td>
<td></td>
<td>204</td>
<td>403</td>
<td>This code is returned if the requester does not have permission to delete the virtual appliance instance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>404</td>
<td>This code is returned if the requested virtual appliance instance is not defined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
</tbody>
</table>

A virtual appliance instance has the following attributes:
created
  Specifies the creation time of the virtual appliance instance, represented as
  the number of milliseconds since midnight, January 1, 1970 UTC.

currentmessage
  Specifies the message associated with the current status of the virtual
  appliance instance. This field contains an 8 character string value that is
  generated by the product.

currentstatus
  Specifies a string constant representing the current status of the virtual
  appliance instance. This field contains an 8 character string value that is
  generated by the product.

currentstatus_text
  Specifies the textual representation of currentstatus. This is a string
  representation of currentstatus in the preferred language of the requester
  and is automatically generated by the product.

desiredstatus
  Specifies the desired status of the virtual appliance instance. Setting this
  value causes IBM Cloud Orchestrator to initiate the steps needed to get the
  virtual system appliance to this state. This value is an 8 character string
  value and it can be set to one of the following values: RM01006 (started) or
  RM01011 (stopped).

id
  Specifies the ID of the virtual appliance instance. This numeric value is
  automatically generated by the product.

name
  Specifies the display name associated with this virtual appliance instance.
  This string contains a string value with a maximum of 1024 characters.

type
  Is always set to APPLIANCE.

updated
  Specifies the time that the virtual appliance instance was last updated,
  represented as the number of milliseconds since midnight, January 1, 1970
  UTC.

GET /resources/virtualApplianceInstances example
[
  {
    "currentstatus_text": "Stopped",
    "currentmessage": "",
    "name": "My image",
    "desiredstatus": "",
    "ownerid": 40,
    "restartstage": 0,
    "environmentprofileid": null,
    "elasticmessage": "",
    "currentmaintenanceid": null,
    "id": 61,
    "ipasmastergroupuuid": null,
    "deploymentid": null,
    "offeringid": -1,
    "iptype": "IPv4",
    "mode": "",
    "priority": 1,
    "created": 1393239516187,
    "currentstatus": "RM01011",
    "patternid": -1,
    "creator": "admin",
    "currentmessage": null,
    "forcecheckout": "N",
  
}
GET /resources/virtualApplianceInstances/{id} example

{
  "currentstatus_text": "Stopped",
  "elasticstatus": "",
  "name": "My image",
  "desiredstatus": "",
  "ownerid": 40,
  "restartstage": 0,
  "environmentprofileid": null,
  "elasticmessage": "",
  "currentmaintenanceid": null,
  "id": 61,
  "ipamastergroupuuid": null,
  "deploymentid": null,
  "offeringid": -1,
  "iptype": "IPv4",
  "mode": "",
  "priority": 1,
  "created": 1393239516187,
  "currentstatus": "RM01011",
  "patternid": -1,
  "creator": "admin",
  "currentmessage": null,
  "forcecheckout": "N",
  "deploymentstatus": null,
  "type": "APPLIANCE",
  "updated": 139323952497,
  "description": null,
  "stage": 1
},

PUT /resources/virtualApplianceInstances/{id} example

Request JSON:

{
  "desiredstatus": "RM01011"
}

DELETE /resources/virtualApplianceInstances/{id} example

This REST API deletes a virtual appliance instance. This REST API call do not accept any parameter.

Example usage:
DELETE /resources/virtualSystems/47

Related tasks:

“REST API reference” on page 655

The representational state transfer (REST) application programming interface (API) is provided by IBM Cloud Orchestrator.
Virtual applications REST API
You can use the REST API to manage your virtual applications.

The following tasks can be completed using the REST API.

Retrieve all virtual applications
GET /resources/virtualApplications/

Table 167. Retrieve all virtual application details

<table>
<thead>
<tr>
<th>Response content-type</th>
<th>application/json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response code</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:
```
[
  {
    "status": "RUNNING",
    "virtual_system": {
      "id": 3,
      "deployment": "https://10.102.155.72:9444/storehouse/user/deployments/d-09c5ac70-27a3-4c3b-96df-d14a3b23b86c/deployment.json",
      "deployment_name": "Sample JEE web application",
      "appmodel": "https://10.102.155.72:9444/storehouse/user/deployments/d-09c5ac70-27a3-4c3b-96df-d14a3b23b86c/appmodel.json",
      "app_type": "application",
      "app_id": "a-3761fe57-2bda-4f9b-b99c-d2c435d69cb7",
      "start_time": "2011-03-25T17:02:57.878Z",
      "id": "d-09c5ac70-27a3-4c3b-96df-d14a3b23b86c",
      "creator": "u-0",
      "topology": "https://10.102.155.72:9444/storehouse/user/deployments/d-09c5ac70-27a3-4c3b-96df-d14a3b23b86c/topology.json",
      "role_error": false
    },
    "deployment": "https://10.102.155.72:9444/storehouse/user/deployments/d-09c5ac70-27a3-4c3b-96df-d14a3b23b86c/deployment.json",
    "deployment_name": "Sample JEE web application",
    "appmodel": "https://10.102.155.72:9444/storehouse/user/deployments/d-09c5ac70-27a3-4c3b-96df-d14a3b23b86c/appmodel.json",
    "app_type": "application",
    "app_id": "a-3761fe57-2bda-4f9b-b99c-d2c435d69cb7",
    "start_time": "2011-03-25T17:02:57.878Z",
    "id": "d-09c5ac70-27a3-4c3b-96df-d14a3b23b86c",
    "creator": "u-0",
    "topology": "https://10.102.155.72:9444/storehouse/user/deployments/d-09c5ac70-27a3-4c3b-96df-d14a3b23b86c/topology.json",
    "role_error": false
  },
  ...
]
```

Retrieve the virtual applications with filter
GET /resources/virtualApplications/{depl_id}

- Filter "app_type": Filter application patterns with application type.
- filterString can be application, or service (for shared service). If filterString is null or empty, all virtual applications will be returned.
- Filter "patterntype" and "version": Filter virtual application with versioned pattern type, for example, ?patterntype=webapp&version=1.0

Table 168. Retrieve the virtual applications with filter

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/virtualApplications/?patterntype=webapp&amp;version=1.0&amp;app_type=application">https://localhost/resources/virtualApplications/?patterntype=webapp&amp;version=1.0&amp;app_type=application</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response code</td>
<td>200</td>
</tr>
</tbody>
</table>
Table 168. Retrieve the virtual applications with filter (continued)

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:

```
[
  {
    "status": "RUNNING",
    "virtual_system": {
      "id": 3
    },
    "deployment": "https://172.16.33.84:9444/storehouse/user/deployments/d-ecaa2e15-f556-43e8-8e09-8905aebab441/deployment.json",
    "deployment_name": "Sample JEE web application",
    "appmodel": "https://172.16.33.84:9444/storehouse/user/deployments/d-ecaa2e15-f556-43e8-8e09-8905aebab441/appmodel.json",
    "app_type": "application",
    "app_id": "a-9d64b797-3d4a-4332-9e09-858ba709a499",
    "start_time": "2011-05-14T12:42:20.950Z",
    "id": "d-ecaa2e15-f556-43e8-8e09-8905aebab441",
    "creator": "u-0",
    "topology": "https://172.16.33.84:9444/storehouse/user/deployments/d-ecaa2e15-f556-43e8-8e09-8905aebab441/topology.json",
    "role_error": false
  }
]
```

Deploy a virtual application

**POST /resources/applicationPatterns/{app_id}/virtualApplications/**

Table 169. Deploy a virtual application details

<table>
<thead>
<tr>
<th>Request content-type</th>
<th>application/json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Response header location</td>
<td><a href="https://localhost/resources/applicationPatterns/a-de901667-3b5f-44af-b27f-ed8971ee552/virtualApplications/d-7956c64e-0fac-49f2-b04e-efbc131a4cc4">https://localhost/resources/applicationPatterns/a-de901667-3b5f-44af-b27f-ed8971ee552/virtualApplications/d-7956c64e-0fac-49f2-b04e-efbc131a4cc4</a></td>
</tr>
<tr>
<td>Response code</td>
<td>201 Created</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>404</td>
<td>The application specified by {appID} is not found</td>
</tr>
<tr>
<td>412</td>
<td>Precondition failed (unable to deploy, such as template)</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Request body:

```
{
  "deployment_name": "My Virtual Application",
  "cloud_group": "1",
  "ssh_keys": ["ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQClOjYa360x0Y+c9l/
  JGQ97zzy5sU8W4Vz+xM1mkn7v46Q0m5L+D5c/vv14hVj6Oz0bK0bSXYvYzv:
  e0shh7WmctOuiciQ5emoe2eaV01yZqP5vBe9V8amTC1ls+Uv/SXFF7uK1V7gP8hBuB
```

Chapter 10. Reference 773
The "deployment_name" is an optional parameter for virtual application name. By default, it will be the name of application. The "cloud_group" is a required parameter for deployment. The "ssh_keys" is an optional parameter for deployment.

Response example:

```json
{
    "status": "RUNNING",
    "deployment_id": "d-7956c64e-0fac-49f2-b04e-efbc131a4cc4",
    "deployment_name": "db2",
    "app_type": "application",
    "app_id": "a-3761fe57-2bda-4f9b-b90c-d2c435d69cb7",
    "start_time": "2011-03-25T17:02:57.878Z",
    "virtual_system": {
        "id": "1"
    },
    "instances": [
        {
            "status": "RUNNING",
            "master": true,
            "last_update": "2011-03-25T17:11:3.750Z",
            "private_ip": "10.102.165.49",
            "reboot_count": 0,
            "stopped_by": "",
            "volumes": [],
            "start_time": "2011-03-25T17:03:51.654Z",
            "id": "rack9.xdblade32b04.22889.03473",
            "name": "database-db2.11301072577884",
            "roles": [
                {
                    "node": "database-db2.11301072577884",
                    "status": "RUNNING",
                    "last_update": "2011-03-25T17:11:14.840Z",
                    "external_uri": "jdbc:db2://10.102.165.49:50000/mydb:user=appdba;password=FgxmZv47TM8GwJD62Y1;",
                    "id": "database-db2.11301072577884.082"
                }
            ],
            "public_ip": "10.102.165.49"
        }
    ],
    "role_error": false
}
```

**Retrieve virtual application instance status**

GET `/resources/virtualApplications/{deployment_id}`

<table>
<thead>
<tr>
<th>Table 170. Retrieve virtual application instance status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example URL</strong></td>
</tr>
<tr>
<td><strong>Response content-type</strong></td>
</tr>
<tr>
<td><strong>Response code</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Table 170. Retrieve virtual application instance status (continued)

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>404</td>
<td>The application specified by {appID} is not found</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Response example:
```
{
  "status": "RUNNING",
  "deployment_name": "db2",
  "app_type": "application",
  "app_id": "a-3761fe57-2bda-4f9b-b90c-d2c435d69cb7",
  "start_time": "2011-03-25T17:02:57.878Z",
  "virtual_system": {
    "id": "1"
  },
  "instances": [
    {
      "status": "RUNNING",
      "master": true,
      "last_update": "2011-03-25T17:11:3.750Z",
      "private_ip": "10.102.165.49",
      "reboot_count": 0,
      "stopped_by": "",
      "volumes": [ ],
      "start_time": "2011-03-25T17:03:51.654Z",
      "id": "rack9.xblade32b04.22889.03473",
      "name": "database-db2.1130107257788",
      "roles": [
        {
          "node": "database-db2.1130107257788",
          "status": "RUNNING",
          "last_update": "2011-03-25T17:11:4.840Z",
          "external_uri": "jdbc:db2://10.102.165.49:50000/mydb:user=appdba;
password=FgxmZv47TM8GwJD62Y1;",
          "id": "database-db2.1130107257788.022"
        }
      ],
      "public_ip": "10.102.165.49"
    }
  ],
  "role_error": false
}
```

**Update virtual application status**

PUT /resources/virtualApplications/{depl_ID}/

In request body, the values of the operation are stop and terminate.

Table 171. Updated virtual application status details

<table>
<thead>
<tr>
<th>Example URL</th>
<th><a href="https://localhost/resources/virtualApplications/d-7956c64e-0fac-492-b04e-efbc131a4cc4">https://localhost/resources/virtualApplications/d-7956c64e-0fac-492-b04e-efbc131a4cc4</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Request body</td>
<td>{ &quot;operation&quot;: &quot;kill&quot; }</td>
</tr>
<tr>
<td>Response code</td>
<td>202 Accepted</td>
</tr>
<tr>
<td>Response code</td>
<td>401 The user is not authorized to perform this action.</td>
</tr>
</tbody>
</table>
### Table 171. Updated virtual application status details (continued)

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>404</td>
<td>The application specified by {appID} is not found</td>
</tr>
<tr>
<td>409</td>
<td>Conflict</td>
</tr>
<tr>
<td>412</td>
<td>Precondition failed (invalid operation)</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

### Delete a virtual application

DELETE /resources/virtualApplications/{depl_ID}

#### Table 172. Delete a virtual application details

<table>
<thead>
<tr>
<th>Example URL</th>
<th><code>https://localhost/resources/virtualApplications/d-7956c64e-0fac-49f2-b04e-efbc131a4cc4</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Request body</td>
<td><code>{&quot;success&quot;:&quot;true&quot;}</code></td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>Note: If the deployment specified by {depl_id} is not found, a 200 response code returns, response body: <code>{&quot;success&quot;: &quot;false&quot;}</code></td>
<td></td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>409</td>
<td>Conflict (same action is already in process)</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

### Update application access right for the specified user name or group name

PUT /resources/virtualApplications/{appID}/accessRights/{name}?{ user or group }

#### Table 173. Update application access right for the specified user name or group name

<table>
<thead>
<tr>
<th>Example URL</th>
<th><code>https://localhost/resources/virtualApplications/d-7956c64e-0fac-49f2-b04e-efbc131a4cc4/accessRights/Everyone?group</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Response content-type</td>
<td>application/json</td>
</tr>
<tr>
<td>Request body example</td>
<td><code>{ &quot;access_rights&quot;: &quot;F&quot; }</code></td>
</tr>
<tr>
<td>Response code</td>
<td>200 OK</td>
</tr>
<tr>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td>404</td>
<td>The application specified by {appID} is not found.</td>
</tr>
<tr>
<td>412</td>
<td>Invalid parameter supplied, for example, the json file is invalid.</td>
</tr>
</tbody>
</table>
**Virtual Images REST API**

You can use the representational state transfer (REST) application programming interface (API) to manage virtual images.

### Available HTTP Methods

**Table 174. REST API for VirtualMachines**

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/resources/virtualImages</td>
<td>application/json</td>
<td>200</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Returns the list of virtual images that are visible to the client.</td>
<td>This code is returned if the requester does not have access to list virtual images.</td>
</tr>
<tr>
<td>GET</td>
<td>/resources/virtualImages/{id}</td>
<td>application/json</td>
<td>200</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Returns the virtual image associated with the given ID.</td>
<td>This code is returned if the requester does not have access to the requested virtual image.</td>
</tr>
<tr>
<td>GET</td>
<td>/resources/clouds/{cloud_id}/images</td>
<td>application/json</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Returns the list of images of the selected cloud group. See the example for a sample of the data returned.</td>
<td>This code is returned if the IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
</tbody>
</table>
Table 174. REST API for VirtualMachines (continued)

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/resources/clouds/ [cloud_id]/images/ [image_id]</td>
<td>application/json</td>
<td>200 Returns the selected image of the selected cloud group. If you specify a .ovf suffix the response is an OVF XML file. See the example for a sample of the data returned.</td>
<td>500 This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
<tr>
<td>GET</td>
<td>/resources/clouds/ [cloud_id]/images/ [image_id].ovf</td>
<td>application/json</td>
<td>200 Returns the list of virtual images that are visible to the client.</td>
<td>403 This code is returned if the requester does not have access to list virtual images.</td>
</tr>
<tr>
<td>POST</td>
<td>/resources/templates</td>
<td>application/json</td>
<td>201 The resources has been created successfully.</td>
<td>500 This code is returned if the IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
</tbody>
</table>

A virtual image has the following attributes:

**build**  Specifies the build number associated with the virtual image. This string value is supplied by the provider of the virtual image and cannot be changed.

**created**  Specifies the creation time of the virtual image, represented as the number of milliseconds since midnight, January 1, 1970 UTC. The value is numeric and is automatically generated by the product.

**currentmessage**  Specifies the message associated with the current status of the virtual image. This field contains an 8 character string value that is generated by the product.
currentmessage_text
Specifies the textual representation of currentmessage in the preferred
language of the requester and is automatically generated by the product.

currentstatus
Specifies a string constant representing the current status of the virtual
image. This field contains an 8 character string value that is generated by
the product.

currentstatus_text
Specifies the textual representation of currentstatus in the preferred
language of the requester and is automatically generated by the product.

description
Specifies the description of the virtual image. This string value is supplied
by the provider of the virtual image and cannot be changed.

hypervisortype
Specifies OpenStack as the type of hypervisor on which this virtual image
can run.

id
Specifies the ID of the virtual image. This numeric value is automatically
generated by the product.

linked
Specifies 1 (true) if the image is registered from an external hypervisor, 0
(false) if the image has been imported locally. This is a read-only value.

linked_cloudid
Specifies the ID of the cloud group from which the image has been
registered. This is a read-only value and it is set only in the linked
attribute is set to 1 (true).

linked_entityid
Specifies the ID of the image on the external hypervisor. This is a read-only
value and it is set only in the linked attribute is set to 1 (true).

name
Specifies the name of the virtual image. This string value is supplied by
the provider of the virtual image and cannot be changed.

operatingsystemdescription
Specifies a textual description of the operating system contained within the
virtual image. The string value for this optional attribute is supplied by the
provider of the virtual image and cannot be changed.

operatingsystemid
Specifies the numeric ID of the operating system contained within the
virtual image. The ID is one of the values described in the common
information model (CIM) specification and is supplied by the provider of
the virtual image.

operatingsystemid_text
Specifies a textual representation of operatingsystemid.

operatingsystemversion
Specifies the version of the operating system contained within the virtual
image. The string value for this optional attribute is supplied by the
provider of the virtual image and cannot be changed.

owner
Specifies the uniform resource identifier (URI) of the user that owns this
virtual image. The URI is relative and should be resolved against the URI
of the virtual image.
updated

Specifies the time that the virtual image was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

version

Specifies the version of the virtual image. This string value is supplied by the provider of the virtual image and cannot be changed.

GET /resources/virtualImages example

```json
[
  {
    "currentstatus_text": "Draft",
    "name": "My SuSE Small",
    "operatingsystemid": 84,
    "operatingsystemdescription": "Suse Linux Enterprise 10 (32-bit)",
    "hypervisortype": "OpenStack",
    "version": "0",
    "linked": 1,
    "id": 1,
    "currentmessage_text": null,
    "operatingsystemid_text": "SLES",
    "linked_entityid": "1-vm-76",
    "created": 1339571308504,
    "operatingsystemversion": "10",
    "linked_cloudid": 1,
    "currentstatus": "RM01027",
    "currentmessage": "",
    "build": "",
    "owner": "/resources/users/1",
    "updated": 1339575604458,
    "description": "A virtual machine"
  }
]
```

GET /resources/virtualImages/{id} example

```json
{
  "currentstatus_text": "Draft",
  "name": "My SuSE Small",
  "operatingsystemid": 84,
  "operatingsystemdescription": "Suse Linux Enterprise 10 (32-bit)",
  "hypervisortype": "OpenStack",
  "version": "0",
  "linked": 1,
  "id": 1,
  "currentmessage_text": null,
  "operatingsystemid_text": "SLES",
  "linked_entityid": "2496c17c-c303-4dd1-9008-0d16f8161b9c",
  "created": 1339571308504,
  "operatingsystemversion": "10",
  "linked_cloudid": 1,
  "currentstatus": "RM01027",
  "currentmessage": "",
  "build": "",
  "owner": "/resources/users/1",
  "updated": 1339575604458,
  "description": "A virtual machine"
}
```

GET /resources/clouds/{id}/images example

```json
[
  {
    "architecture": "x86",
    ...
  }
]
```
"description": "Microsoft Windows Server 2008 R2 (64-bit)",
"hypervisor": "VMware",
"id": "S1_1-vm-106",
"name": "W2K8_R2-vmx4-CCS-template",
"repository": "S1_1",
"version": "vmx-04",
"linked": "false",
"published": "false"
},

{ "architecture": "x86",
"description": "Red Hat Enterprise Linux 5 (64-bit)",
"hypervisor": "VMware",
"id": "S1_1-vm-43",
"name": "JWeMaestroTemplate",
"repository": "S1_1",
"version": "vmx-04",
"linked": "true",
"templateId": 17,
"published": "true"
}.

GET /resources/clouds/{id}/images/{image_id} example

{
 "architecture": "x86",
"description": "Red Hat Enterprise Linux 5 (64-bit)",
"hypervisor": "VMware",
"id": "S1_1-vm-43",
"name": "JWeMaestroTemplate",
"repository": "S1_1",
"version": "vmx-04",
"linked": "true",
"templateId": 17,
"published": "true"
}

GET /resources/templates example

[ {
 "currenteditionid": 1,
 "advancedoptionsdescription": null,
 "name": "My SuSE Small",
 "operatingsystemid": 84,
 "operatingsystemdescription": "Suse Linux Enterprise 10 (32-bit)",
 "ownerid": 1,
 "version": "0",
 "linked": 1,
 "id": 1,
 "operatingsystemid_text": "SLES",
 "hasadvancedoptions": "F",
 "advancedoptionsaccepted": "F",
 "linked_entityid": "2496c17c-c303-4dd1-9008-0d16f816c9ac",
 "hardware": {
 "memory": 192,
 "vcpu": 2,
 "diskDetails": [
 { "size": "1572864000",
 "hardwareid": 1,
 "label": "vmdisk1",
 "filename": "SLES11_32_Small.vmdk",
 "created": 1339571317752,
 "updated": 1339571317752,
 "id": 1
 }]

Chapter 10. Reference  781
POST /resources/templates example

Request JSON:
{
   "url": "SampleImageHost/images/SampleImage.ova"
}

where url is the address from which the image can be downloaded or a path to the locally stored OVA file.

Virtual machines REST API
You can use the representational state transfer (REST) application programming interface (API) to manage virtual machines.
## Available HTTP methods

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/resources/virtualSystems/ [id]/virtualMachines</td>
<td>application/json</td>
<td>200</td>
<td>Returns the list of virtual system instances that are visible to the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>403</td>
<td>This code is returned if the requester does not have access to list virtual machines on the virtual system instance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
<tr>
<td>GET</td>
<td>/resources/virtualSystems/ [id]/virtualMachines/[id]</td>
<td>application/json</td>
<td>200</td>
<td>Returns the virtual machine associated with the given ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>403</td>
<td>This code is returned if the requester does not have access to the requested virtual system instance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>404</td>
<td>This code is returned if the requested virtual machine is not defined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
</tbody>
</table>
Table 175. REST API for virtual machines (continued)

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/resources/virtualSystems/</td>
<td>application/json</td>
<td>201</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>[id]/virtualMachines</td>
<td></td>
<td>The virtual machine(s) have been defined in IBM Cloud Orchestrator. The virtual machines are started when the product is able to do so. The response body contains a list of URIs for the new virtual machines. Relative URIs are resolved against the URI used for this request. The URI of the first virtual machine is also included in the HTTP Location header of the response.</td>
<td>This code is returned if there are problems parsing the JSON data in the request.</td>
</tr>
<tr>
<td>PUT</td>
<td>/resources/adoptUnmanagedVM</td>
<td>application/json</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The virtual system instance was successfully updated. The response body is empty.</td>
<td>This code is returned if there are problems parsing the JSON data in the request.</td>
</tr>
</tbody>
</table>

A virtual system instance has the following attributes:
cloud Specifies the uniform resource identifier (URI) of the cloud group in which this virtual machine was started. The URI is relative and should be resolved against the URI of the virtual machine.

cpucount Specifies the number of virtual CPUs assigned to this virtual machine. This value is an integer.

created Specifies the creation time of the virtual machine, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

currentmessage Specifies the message associated with the current status of the virtual machine. This field contains an 8 character string value that is generated by the product.

currentmessage_text Specifies the textual representation of currentmessage. This is a string representation of currentmessage in the preferred language of the requester and is automatically generated by the product.

currentstatus Specifies a string constant representing the current status of the virtual machine. This field contains an 8 character string value that is generated by the product.

currentstatus_text Specifies the textual representation of currentstatus. This is a string representation of currentstatus in the preferred language of the requester and is automatically generated by the product.

displayname Specifies the display name used on the hypervisor for this virtual machine. This field contains a string value with a maximum of 1024 characters.

hypervisor Specifies the URI of the hypervisor on which this virtual machine is running. The URI is relative and should be resolved against the URI of the virtual machine.

hypervisormachineid Specifies the ID assigned to the virtual machine by the hypervisor. This field contains a string value with a maximum of 1024 characters.

id Specifies the ID of the virtual machine. This value is numeric and is automatically generated by the product.

nics Specifies a list of the URIs of the IP addresses used by this virtual machine. URIs are relative and resolved against the URIs of the virtual machine.

memory Specifies the amount of memory allocated to this virtual machine, represented in megabytes. This value is an integer.

name Specifies the display name associated with this virtual machine. This field contains a string value with a maximum of 1024 characters.

runtimeid Specifies the runtime ID generated by the hypervisor on which this virtual machine is running. This field contains a string value with a maximum of 1024 characters.
storageid
Specifies the hypervisor storage ID of the storage on which this virtual machine resides. This field contains a string value with a maximum of 1024 characters.

updated
Specifies the time the virtual system instance was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

GET /resources/virtualSystems/{id}/virtualMachines example
[
  {
    "cloud": "/resources/clouds/1",
    "cpucount": 1,
    "created": 1245376939141,
    "currentmessage": "RM07044",
    "currentmessage_text": "Virtual machine has been started",
    "currentstatus": "RM01006",
    "currentstatus_text": "Started",
    "displayname": "two vm virtual system instance vm-009-238 dmgr",
    "hypervisor": "/resources/hypervisors/1",
    "hypervisormachineid": "https://virtualwas.rtp.raleigh.ibm.com/sdk#HostSystem#ha-host",
    "id": 8,
    "nics": [
      {
        "ip": "/resources/ipGroups/1/ips/1",
        "ip_hostname": "test1",
        "ip_address": "192.168.1.100"
      }
    ],
    "memory": 2048,
    "name": "DMGR 1",
    "runtimeid": "https://virtualwas.rtp.raleigh.ibm.com/sdk#VirtualMachine#29424",
    "storageid": "https://virtualwas.rtp.raleigh.ibm.com/sdk#Datastore#49824cfe-4bd840fb-a97b-001a643670e6",
    "updated": 1245377403612
  },
  {
    "cloud": "/resources/clouds/1",
    "cpucount": 1,
    "created": 1245376954282,
    "currentmessage": "RM07035",
    "currentmessage_text": "Waiting for initialization to complete",
    "currentstatus": "RM01005",
    "currentstatus_text": "Starting",
    "displayname": "two vm virtual system instance vm-009-239 custom",
    "hypervisor": "/resources/hypervisors/1",
    "hypervisormachineid": "https://virtualwas.rtp.raleigh.ibm.com/sdk#HostSystem#ha-host",
    "id": 9,
    "ip": "/resources/ipGroups/1/ips/2",
    "memory": 2048,
    "name": "Custom Node 3",
    "runtimeid": "https://virtualwas.rtp.raleigh.ibm.com/sdk#VirtualMachine#29440",
    "storageid": "https://virtualwas.rtp.raleigh.ibm.com/sdk#Datastore#49824d1c-f0c6ea83-f3e0-001a643670e6",
    "updated": 1245377415829
  }
]

See the description of GET /resources/virtualSystems/{id}/virtualMachines/{id} for attribute details.
GET /resources/virtualSystems/{id}/virtualMachines/{id} example

```
{
  "cloud": "/resources/clouds/1",
  "cpucount": 1,
  "created": 1245376939141,
  "currentstatus_text": "Started",
  "currentmessage": "RM07044",
  "currentmessage_text": "Virtual machine has been started",
  "currentstatus": "RM01006",
  "displayname": "two vm virtual system instance vm-009-238 dmgr",
  "hypervisor": "/resources/hypervisors/1",
  "hypervisormachineid": "https://virtualwas.rtp.raleigh.ibm.com/
    sdk#HostSystem#ha-host",
  "id": 8,
  "ip": "/resources/ipGroups/1/ips/1",
  "memory": 2048,
  "name": "DMGR 1",
  "runtimeid": "https://virtualwas.rtp.raleigh.ibm.com/
    sdk#VirtualMachine#29424",
  "storageid": "https://virtualwas.rtp.raleigh.ibm.com/
    sdk#Datastore#49824cfe-4bd840fb-a97b-001a643670e6",
  "updated": 1245377675248
}
```

POST /resources/virtualSystems/{id}/virtualMachines example

Request JSON:
```
{
  "virtualmachine": "/resources/virtualSystems/5/virtualMachines/12",
  "desiredcount": 2
}
```

Response JSON:
```
[
  "/resources/virtualSystems/5/virtualMachines/16",
  "/resources/virtualSystems/5/virtualMachines/17"
]
```

PUT /resources/adoptUnmanagedVM example

Request JSON:
```
{ "runtimeids": ["runtimeId1,runtimeId2,..."] }
```

Related tasks:
"REST API reference“ on page 655

The representational state transfer (REST) application programming interface (API) is provided by IBM Cloud Orchestrator.

Virtual system instances REST API
You can use the representational state transfer (REST) application programming interface (API) to manage virtual system instances.
### Available HTTP Methods

*Table 176. REST API for VirtualSystems*

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/resources/virtualSystems</td>
<td>application/json</td>
<td>200</td>
<td>Returns the list of virtual system instances that are visible to the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>403</td>
<td>This code is returned if the requester does not have access to list virtual system instances.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>This code is returned if IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
<tr>
<td>POST</td>
<td>/resources/virtualSystems</td>
<td>application/json</td>
<td>201</td>
<td>The virtual system instance has been created and is included in the response body. The URI of the new virtual system instance is included in the Location header of the response.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400</td>
<td>This code is returned if there are problems parsing the JSON data in the request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>403</td>
<td>This code is returned if the requester does not have permission to create virtual system instances.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>This code is returned if the IBM Cloud Orchestrator encountered an internal error while processing the request.</td>
</tr>
<tr>
<td>HTTP Method</td>
<td>URI Pattern</td>
<td>Data Format</td>
<td>Success Codes</td>
<td>Error Codes</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>GET</td>
<td>/resources/virtualSystems/ [id]</td>
<td>application/json</td>
<td>200</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>404</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>PUT</td>
<td>/resources/virtualSystems/ [id]</td>
<td>application/json</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>403</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>404</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>
Table 176. REST API for VirtualSystems (continued)

<table>
<thead>
<tr>
<th>HTTP Method</th>
<th>URI Pattern</th>
<th>Data Format</th>
<th>Success Codes</th>
<th>Error Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>/resources/virtualSystems/</td>
<td></td>
<td>204</td>
<td>403</td>
</tr>
<tr>
<td></td>
<td>{id}</td>
<td></td>
<td>The virtual</td>
<td>This code is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>system instance</td>
<td>returned if the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>has been</td>
<td>requester does</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>deleted.</td>
<td>not have</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>permission to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>delete the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>virtual</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>system instance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>404</td>
<td>This code is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>returned if the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>requested</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>virtual</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>system instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is not defined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>This code is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>returned if the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IBM Cloud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Orchestrator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>encountered an</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>internal error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>while processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the request.</td>
</tr>
</tbody>
</table>

The following attributes are required to create a virtual system:

**endtime**
- Specifies the time the virtual system instance is to be stopped, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This attribute is optional. If not specified, the virtual system instance will run until it is manually stopped.

**environmentProfile**
- This attribute specifies the URI of the environment in which to deploy the new virtual system instance. The `environmentProfile` attribute must be specified at this level.

**name**
- Specifies the name for the new virtual system instance.

**pattern**
- Specifies the URI of the pattern to be used for the new virtual system.

**virtualimage**
- Specifies the URI of the virtual image to be used for single image deployment. ‘X-IBM-Workload-Deployer-API-Version’ in the header must be 4.0 and higher to enable this attribute. Either pattern or virtualimage can be used as attributes.

**parts**
- Specifies a list containing one map per part contained in the pattern.

Because you are using the `environmentProfile` attribute, then you also use one of the cloud and ipGroup pairs provided in the environment profile. The environment profile provides the valid cloud and IP group attributes for that profile:

**cloud**
- Specifies the URI of the cloud in which to deploy the new virtual system.
ipGroup
Specifies the URI of the IP Group in which to deploy the new virtual system.

ipAddress
Specifies the IP address of the virtual machine in the virtual system instance. This attribute is required when the value of IP addresses provided by in the environment profile is pattern deployer.

Important: If the environment profile indicates that the pattern deployer provides the IP address, then you cannot specify an IP address that is contained within the IP groups that are already defined in IBM Cloud Orchestrator.

hostname
Specifies the hostname of the virtual machine in the virtual system instance. This attribute is optional when the value of IP addresses provided by in the environment profile is pattern deployer.

The following example shows how the ipGroup, ipAddress, and hostname attributes might be specified:

"nics": [{
  "ipGroup": "/resources/ipGroups/1",
  "hostname": "myhostname.mycompany.com",
  "ipAddress": "1.1.1.2"
},
{
  "ipGroup": "/resources/ipGroups/1",
  "hostname": "myhostname1.mycompany.com",
  "ipAddress": "1.1.1.3"
}]

The map for each part contains the following attributes:

id Specifies the ID of the pattern part. This numeric value is automatically generated by the product.

label Specifies displayable text used to identify the part.

description Specifies a textual description of the part.

flavor Specifies the predefined size of the part in terms of CPU and memory.

properties Specifies a list containing one map per property defined for the part.

scripts Specifies a list containing one map per script defined for the part.

The map for each part property contains the following attributes:

description Specifies a textual description of the property.

key Specifies a string that uniquely identifies the property within the part property.

description Specifies a textual description of the property.

label Specifies displayable text used to identify the property.

pclass Specifies a string value used to identify related properties within a part. The combination of pclass and key values is unique for every property contained in a given part.
type Specifies a string indicating the type of values that can be assigned to this property. The value will be one of "string", "integer" or "boolean".

validValues For properties that are only allowed to have certain values, the validValues attribute contains a list of the allowable values.

class

value Specifies the default value for the property. The type of this value depends on the property’s type.

The map for each script contains the following attributes:

description Specifies a textual description of the script.

id Specifies the ID of the pattern script. This numeric value is automatically generated by the product.

label Specifies the displayable text used to identify the script.

parameters Specifies a list containing one map per parameter defined for the script.

The map for each parameter contains the following attributes:

key Specifies a string that uniquely identifies the parameter. The script key is a string with a maximum length of 4098 characters.

value The default value for the parameter. All parameter have string values with a maximum length of 4098 characters.

starttime Specifies the time the virtual system instance is to be started, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This attribute is optional. If not specified, the virtual system instance starts as soon as possible.

**GET /resources/virtualSystems example**

```
[
  {
    "created": 1245041940730,
    "currentmessage": null,
    "currentmessage_text": null,
    "currentstatus": "RM01036",
    "currentstatus_text": "Queued",
    "desiredstatus": "",
    "desiredstatus_text": null,
    "id": 6,
    "name": "futurevs",
    "owner": "/resources/users/1",
    "pattern": "/resources/patterns/1",
    "updated": 1245041940730
  },
  {
    "created": 1245356439153,
    "currentmessage": "RM07045",
    "currentmessage_text": "The virtual system instance has been deployed and is ready to use",
    "currentstatus": "RM01006",
    "currentstatus_text": "Started",
    "desiredstatus": "",
    "desiredstatus_text": null,
    "id": 9
  }
]
```
"name": "test virtual system instance",
"owner": "/resources/users/1",
"pattern": "/resources/patterns/6",
"updated": 1245357249316
}
]

See the description of GET /resources/virtualsystems/[id] for attribute details.

**POST /resources/virtualSystems example**

Request JSON:

```json
{
  "environmentProfile": "/resources/environmentProfiles/1",
  "endtime": 1260000000000,
  "name": "sample virtual system instance",
  "parts": [
    {
      "description": "Deployment manager",
      "label": "Deployment manager",
      "id": 1,
      "cloud": "/resources/clouds/1",
      "flavor": "m1.tiny",
      "ipGroup": "/resources/ipGroups/1",
      "ipaddresses": [
        {
          "hostname": "myhostname.mycompany.com",
          "ipaddress": "192.168.0.100"
        }
      ],
      "properties": [
        {
          "description": "Number of virtual CPUs required",
          "key": "numvcpus",
          "label": "Virtual CPUs",
          "pclass": "HWAttributes",
          "type": "integer",
          "validValues": ["1","2","4"],
          "value": "1"
        },
        {
          "description": "Memory size required in megabytes",
          "key": "memsize",
          "label": "Memory size (MB)",
          "pclass": "HWAttributes",
          "type": "integer",
          "value": "3072"
        },
        {
          "description": "This is the cell name of the profile",
          "key": "cell_name",
          "label": "Cell name",
          "pclass": "ConfigWAS",
          "type": "string",
          "value": "DeployerCell"
        },
        {
          "description": "This is the node name of the profile",
          "key": "node_name",
          "label": "Node name",
          "pclass": "ConfigWAS",
          "type": "string",
          "value": "DeployerNode"
        },
        {
          "description": "List of feature packs",
          "key": "augment_list",
          "label": "Feature packs",
          "pclass": "ConfigWAS",
          "type": "string"
        }
      ]
    }
  ]
}
```
"validValues": ["sca","none"],
"value": "none"
],
{
"description": "This is the root password for the system",
"key": "password",
"label": "Password (root)",
"pclass": "ConfigPWD_ROOT",
"type": "string",
"value": "root-password"
},
{
"description": "This is the password for the system and
WebSphere account (virtuser)",
"key": "password",
"label": "Password (virtuser)",
"pclass": "ConfigPWD_USER",
"type": "string",
"value": "virtuser-password"
}],
"scripts": [{
"description": "Test script",
"id": 1,
"label": "test script",
"parameters": [{
"key": "key1",
"value": "value1"
},
{
"key": "key2",
"value": "my value2"
}]
}]
},
{
"description": "Custom nodes",
"id": 3,
"label": "Custom nodes",
"cloud": "/resources/clouds/1",
"flavor": "m1.tiny",
"ipGroup": "/resources/ipGroup/1",
"ipaddresses": [{
"hostname": "myhostname.mycompany.com",
"ipaddress": "192.168.0.102"
}],
"properties": [{
"description": "Number of virtual CPUs required",
"key": "numvcpus",
"label": "Virtual CPUs",
"pclass": "HWAttributes",
"type": "integer",
"validValues": ["1","2","4"],
"value": "1"
},
{
"description": "Memory size required in megabytes",
"key": "memsize",
"label": "Memory size (MB)",
"pclass": "HWAttributes",
"type": "integer",
"value": "2048"
},
{
"description": "This is the cell name of the profile",
"key": "cell_name",
"label": "Cell name",
"pclass": "ConfigWAS",
"type": "string",
"value": "cell-name"
}]}
"type": "string",
"value": "DeployerCell"
},

{"description": "This is the node name of the profile",
"key": "node_name",
"label": "Node name",
"pclass": "ConfigWAS",
"type": "string",
"value": "DeployerNode"
},

{"description": "This is the root password for the system",
"key": "password",
"label": "Password (root)",
"pclass": "ConfigPWD_ROOT",
"type": "string",
"value": "root-password"
},

{"description": "This is the password for the system and WebSphere account (virtuser)",
"key": "password",
"label": "Password (virtuser)",
"pclass": "ConfigPWD_USER",
"type": "string",
"value": "virtuser-password"
}],

"scripts": []
},

"pattern": "/resources/patterns/1",
"starttime": 1250000000000
}

Response JSON:
{
"created": 1245361773378,
"currentmessage": null,
"currentmessage_text": null,
"currentstatus": "RM01036",
"currentstatus_text": "Queued",
"desiredstatus": "",
"desiredstatus_text": null,
"id": 13,
"name": "sample virtual system instance",
"owner": "/resources/users/1",
"pattern": "/resources/patterns/1",
"updated": 1245361773378
}

GET /resources/virtualSystems/[id] example
{
"created": 1245361773378,
"currentmessage": null,
"currentmessage_text": null,
"currentstatus": "RM01036",
"currentstatus_text": "Queued",
"desiredstatus": "",
"desiredstatus_text": null,
"id": 13,
"name": "sample virtual system instance",
"owner": "/resources/users/1",
"pattern": "/resources/patterns/1",
"updated": 1245361773378
}
Note: Key-value pairs that are only used by user interface clients are optional.

A virtual system instance has the following attributes:

**created**
Specifies the creation time of the virtual system instance, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

**currentmessage**
Specifies the message associated with the current status of the virtual system instance. This is an 8 character string value that is generated by the product.

**currentmessage_text**
Specifies the textual representation of currentmessage. This is a string representation of currentmessage in the preferred language of the requester and is automatically generated by the product.

**currentstatus**
Specifies a string constant representing the current status of the virtual system instance. This is an 8 character string value is automatically generated by the product.

**currentstatus_text**
Specifies the textual representation of currentstatus. This is a string representation of currentstatus in the preferred language of the requester and is automatically generated by the product.

**desiredstatus**
Specifies the desired status of the virtual system instance. Setting this value causes IBM Cloud Orchestrator to initiate whatever steps are needed to get the virtual system instance to this state. This value is an 8 character string value that can only be set to the following values: 'RM01006' (started) or 'RM01011' (stopped), 'RM01020' (snapshot).

**desiredstatus_text**
Specifies the textual representation of desiredstatus. This is a string representation of desiredstatus in the preferred language of the requester and is automatically generated by the product.

**id**
Specifies the ID of the virtual system instance. This value is numeric and is automatically generated by the product.

**name**
Specifies the display name associated with this virtual system instance. This field contains a string value with a maximum of 1024 characters.

**owner**
Specifies the URI of the user that owns this virtual system instance. The URI is relative and should be resolved against the URI of the owner.

**pattern**
Specifies the URI of the pattern used to create this virtual system instance. The URI is relative and should be resolved against the URI of the pattern.

**updated**
Specifies the time the virtual system instance was last updated, represented as the number of milliseconds since midnight, January 1, 1970 UTC. This value is numeric and is automatically generated by the product.

**PUT /resources/virtualSystems/{id} example**

Request JSON:
DELETE /resources/virtualSystems example

This REST API deletes a virtual system instance and frees the resources it currently consumes. This REST API call accepts the following single optional query parameter:

delete-virtual-system-force
  A boolean value that indicates the action the product takes if errors are encountered while deleting the virtual system instance. Valid values are true or false. If the query parameter is omitted or is assigned the false value, any errors encountered while deleting the virtual system instance cause the HTTP request to fail with a 500 error code. If the query parameter is assigned the true value, errors are ignored and the deleting processing continues.

Example usage:
DELETE /resources/virtualSystems/47?delete-virtual-system-force=true

Related tasks:
“REST API reference” on page 655
The representational state transfer (REST) application programming interface (API) is provided by IBM Cloud Orchestrator.
Virtual system patterns REST API
You can use the REST API to manage your virtual system patterns.

Retrieve a list of all virtual system patterns
GET /resources/virtualSystemPatterns

Table 177. Retrieve a list of all virtual system patterns

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URL</td>
<td><a href="https://localhost/resources/virtualSystemPatterns">https://localhost/resources/virtualSystemPatterns</a></td>
<td></td>
</tr>
<tr>
<td>Response content-type</td>
<td>application/json</td>
<td></td>
</tr>
</tbody>
</table>
### Table 177. Retrieve a list of all virtual system patterns (continued)

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Response example</strong></td>
<td>Response body:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>[]</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>{ &quot;app_mgmtserver_url&quot;: &quot;https://1xx.0.0.1:9443/services/applications/a-1326e132-5d9e-4830-86d3-1ccbb72b29c46&quot;, &quot;last_modifier&quot;: &quot;admin&quot;, &quot;app_type&quot;: &quot;application&quot;, &quot;app_storehouse_base_url&quot;: &quot;https://1xx.0.0.1:9443/storehouse/user/applications/a-1326e132-5d9e-4830-86d3-1ccbb72b29c46/&quot;, &quot;patterntype&quot;: &quot;vsys&quot;, &quot;app_name&quot;: &quot;BaseImageWithScalingPolicy&quot;, &quot;creator&quot;: &quot;admin&quot;, &quot;version&quot;: &quot;1.0&quot;, &quot;patternversion&quot;: &quot;1.0&quot;, &quot;last_modified&quot;: &quot;2014-02-20T20:15:09Z&quot;, &quot;description&quot;: &quot;&quot;, &quot;create_time&quot;: &quot;2014-02-20T20:15:09Z&quot;, &quot;content_md5&quot;: &quot;c63f76b66b35a718a79b795aaf4f7b1f60f425cf66c11ef8bd4039d5084397fb0c8823369b97336e0735be0640dd581936783286&quot;, &quot;app_id&quot;: &quot;a-1326e132-5d9e-4830-86d3-1ccbb72b29c46&quot;, &quot;access_rights&quot;: { &quot;admin&quot;: &quot;F&quot;, &quot;_group_:Everyone&quot;: &quot;R&quot; }, &quot;content_type&quot;: &quot;application/json&quot; },</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>{ &quot;app_mgmtserver_url&quot;: &quot;https://1xx.0.0.1:9443/services/applications/a-14415000-1eb6-4c41-af91-af06d0928296&quot;, &quot;last_modifier&quot;: &quot;admin&quot;, &quot;app_type&quot;: &quot;application&quot;, &quot;app_storehouse_base_url&quot;: &quot;https://1xx.0.0.1:9443/storehouse/user/applications/a-14415000-1eb6-4c41-af91-af06d0928296/&quot;, &quot;patterntype&quot;: &quot;vsys&quot;, &quot;app_name&quot;: &quot;was&quot;, &quot;creator&quot;: &quot;admin&quot;, &quot;version&quot;: &quot;1.0&quot;, &quot;patternversion&quot;: &quot;1.0&quot;, &quot;last_modified&quot;: &quot;2014-02-20T20:15:09Z&quot;, &quot;description&quot;: &quot;&quot;, &quot;create_time&quot;: &quot;2014-02-20T20:15:09Z&quot;, &quot;content_md5&quot;: &quot;8f5d03d74803a367ec85db0690274fb058d410c1b4ca8d117ccf5c742b4cd9f8b750d7d3ef5073af691190dcb1653f84e7&quot;, &quot;app_id&quot;: &quot;a-14415000-1eb6-4c41-af91-af06d0928296&quot;, &quot;access_rights&quot;: { &quot;admin&quot;: &quot;F&quot;, &quot;_group_:Everyone&quot;: &quot;R&quot; }, &quot;content_type&quot;: &quot;application/json&quot; },</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>{ &quot;app_mgmtserver_url&quot;: &quot;https://1xx.0.0.1:9443/services/applications/a-7cc55c0b-5656-4e79-93ad-d6935dea71b3&quot;, &quot;last_modifier&quot;: &quot;admin&quot;, &quot;app_type&quot;: &quot;application&quot;, &quot;app_storehouse_base_url&quot;: &quot;https://1xx.0.0.1:9443/storehouse/user/applications/a-7cc55c0b-5656-4e79-93ad-d6935dea71b3/&quot;, &quot;patterntype&quot;: &quot;vsys&quot;, &quot;app_name&quot;: &quot;Database&quot;, &quot;creator&quot;: &quot;admin&quot;, &quot;version&quot;: &quot;1.0&quot;, &quot;patternversion&quot;: &quot;1.0&quot;, &quot;last_modified&quot;: &quot;2014-02-20T20:15:09Z&quot;, &quot;description&quot;: &quot;&quot;, &quot;create_time&quot;: &quot;2014-02-20T20:15:09Z&quot;, &quot;content_md5&quot;: &quot;398c13c39e49ca8eede91141f4c7a5924652d9525987c11ca148f3069e6f8ce89830c53fe089e72f6ff4ffcc2f0db527c7b9d3a3&quot;, &quot;app_id&quot;: &quot;a-7cc55c0b-5656-4e79-93ad-d6935dea71b3&quot;, &quot;access_rights&quot;: { &quot;admin&quot;: &quot;F&quot;, &quot;_group_:Everyone&quot;: &quot;R&quot; }, &quot;content_type&quot;: &quot;application/json&quot; },</code></td>
<td></td>
</tr>
</tbody>
</table>
**Return a specific virtual system pattern**

GET /resources/virtualSystemPatterns/pattern ID

*Table 178. Return a specific virtual system pattern*

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URL</td>
<td><a href="https://localhost/resources/virtualSystemPatterns/a-1326e132-5d9e-4830-86d3-1ccb72b29c46">https://localhost/resources/virtualSystemPatterns/a-1326e132-5d9e-4830-86d3-1ccb72b29c46</a></td>
<td></td>
</tr>
<tr>
<td>Response content-type</td>
<td>application/json</td>
<td></td>
</tr>
</tbody>
</table>
### Table 178. Return a specific virtual system pattern (continued)

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response example</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Response body:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;app_mgmtserver_url&quot;: &quot;<a href="https://1xx.0.0.1:9443/services/applications/a-9af3ab60-c60c-4e9e-9f3a-1c31543becde">https://1xx.0.0.1:9443/services/applications/a-9af3ab60-c60c-4e9e-9f3a-1c31543becde</a>&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;model&quot;: {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;nodes&quot;: [</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;id&quot;: &quot;OS Node&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;attributes&quot;: {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;ConfigPWD_USER.password&quot;: &quot;&lt;xor&gt;LzsLChvLTs=&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;HWAttributes.memsizes&quot;: 2048,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;HWAttributes.numvcpus&quot;: 1,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;ConfigPWD_ROOT.password&quot;: &quot;&lt;xor&gt;LzsLChvLTs=&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;type&quot;: &quot;image:OS Node:IBM OS Image for Red Hat Linux Systems:2.0.0.4:148&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;description&quot;: &quot;&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;name&quot;: &quot;mytestpattern&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;app_type&quot;: &quot;application&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;patterntype&quot;: &quot;vsys&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;links&quot;: [</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;locked&quot;: false,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;readonly&quot;: false,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;patternversion&quot;: &quot;1.0&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;version&quot;: &quot;1.0&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;last_modifier&quot;: &quot;admin&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;app_type&quot;: &quot;application&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;app_storehouse_base_url&quot;: &quot;<a href="https://1xx.0.0.1:9444/storehouse/user/applications/a-9af3ab60-c60c-4e9e-9f3a-1c31543becde/">https://1xx.0.0.1:9444/storehouse/user/applications/a-9af3ab60-c60c-4e9e-9f3a-1c31543becde/</a>&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;layers&quot;: [</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;id&quot;: &quot;layer&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;nodes&quot;: [</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;OS Node&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;patterntype&quot;: &quot;vsys&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;tooling&quot;: {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;nodes&quot;: [</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;id&quot;: &quot;OS Node&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;location&quot;: {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;y&quot;: &quot;141px&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;x&quot;: &quot;349px&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;ismini&quot;: false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;links&quot;: [</td>
<td></td>
<td></td>
</tr>
<tr>
<td>]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>],</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;app_name&quot;: &quot;mytestpattern&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;version&quot;: &quot;1.0&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;patternversion&quot;: &quot;1.0&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;creator&quot;: &quot;admin&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;content_sha2&quot;: &quot;d491ba6a605f5138e6be87f8efdaa849e5b3bc0de5522e7577bb6e3be18e964f6fc16ef12f613ef419e22fdd081e84fd335ee9544a4d&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;last_modified&quot;: &quot;2014-02-19T20:54:28Z&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;description&quot;: &quot;&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;create_time&quot;: &quot;2014-02-19T20:54:28Z&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;content_md5&quot;: &quot;423b79a64893a46817e14cd223fa80aa&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;access_rights&quot;: {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;admin&quot;: &quot;F&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>},</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;app_id&quot;: &quot;a-9af3ab60-c60c-4e9e-9f3a-1c31543becde&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;locked&quot;: &quot;false&quot;,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;content_type&quot;: &quot;application/json&quot;,</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Create a virtual system pattern from an existing virtual system pattern (clone)

POST /resources/virtualSystemPatterns/?source={app_id}&app_name={name}&app_type={app_type}

A unique ID is generated for the virtual system.
- Attribute "source": specify the ID for the existing virtual system pattern (required)
- Attribute "app_name": specify the virtual system pattern name (required)
- Attribute "app_type": specify the type for target application. The values can be application or template. The default value is application.

Table 179. Create a virtual system pattern from an existing virtual system pattern (clone)

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URL</td>
<td><a href="https://localhost/resources/virtualSystemPatterns/">https://localhost/resources/virtualSystemPatterns/</a> ?source=a-679a68f4-6798-424f-8039-1f682f949f45 &amp;app_name=testSys</td>
<td>Create a virtual system named testSys from application with ID a-679a68f4-6798-424f-8039-1f682f949f45</td>
</tr>
<tr>
<td>Response content-type</td>
<td>application/json</td>
<td></td>
</tr>
<tr>
<td>Response header location</td>
<td><a href="https://localhost/resources/virtualSystemPatterns/a-fb70796e-1b13-467a-babe-b8b700bd563b">https://localhost/resources/virtualSystemPatterns/a-fb70796e-1b13-467a-babe-b8b700bd563b</a></td>
<td></td>
</tr>
<tr>
<td>Request content-type</td>
<td>application/json</td>
<td></td>
</tr>
<tr>
<td>Response headers content-type</td>
<td>application/json</td>
<td></td>
</tr>
<tr>
<td>Response body</td>
<td>JSON format of the pattern</td>
<td></td>
</tr>
<tr>
<td>Response code</td>
<td>201 Created</td>
<td></td>
</tr>
<tr>
<td></td>
<td>403 Access forbidden</td>
<td></td>
</tr>
<tr>
<td></td>
<td>404 Not found</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500 Unexpected error</td>
<td></td>
</tr>
</tbody>
</table>

Deploy a virtual system pattern

There are two options when you deploy a virtual system pattern:
1. Deploy the virtual system pattern with the placement that is determined by IBM Cloud Orchestrator, if applicable, or by not using placement if it is not supported by the virtual system pattern. To deploy the virtual system pattern with this method, use the POST REST API and do not include the placement_only parameter in the request body, or set it to False.
2. Modify the placement before you deploy the virtual system pattern, and use the modified placement for the deployment. To use this method, the deployment must be called in two phases:
• First, generate the placement and topology by including `placement_only:` "True" in the request body.
  This parameter tells the system to generate a placement for the deployment, which is returned in response body. You can modify this placement before you pass it to the system in the second phase to deploy the virtual system pattern.

• Then, deploy the virtual system pattern by calling the PUT REST API with the `deployPlacement` operation, and pass the modified placement for the deployment in the request body.

**Note:**
• Because placement is handled by the system, you do not have to specify a cloud group or IP group if an environment profile is specified. If the pattern cannot use placement, then the cloud group and IP group parameters are required. For example, if some of the plug-ins in the pattern do not require Foundation 2.1, the application cannot use placement. In this scenario, the cloud group and IP group are required. If you do not specify these parameters for a pattern that cannot use placement, the deployment fails.

• If "placement_only": True is in the request body, but placement is not supported for the pattern, that parameter is ignored by the system. The pattern is deployed as if the `placement_only` was not specified, or was set to False.

**Restriction:** If your API version is not 5.0.0.0:
• The cloud group and IP group are required parameters.
• Deploying to an environment profile with the cloud management type set to "By way of external network" is not supported.

Deploy the virtual system pattern, or generate the placement if you are using the two-phase method:

```plaintext
POST /resources/virtualSystemPatterns/pattern ID/virtualSystemInstances/
```

**Table 180. Deploy a virtual system pattern**

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URL</td>
<td><a href="http://server/resources/virtualSystemPatterns/a-123/virtualSystemInstances/">http://server/resources/virtualSystemPatterns/a-123/virtualSystemInstances/</a></td>
<td></td>
</tr>
<tr>
<td>Request content-type</td>
<td>application/json</td>
<td></td>
</tr>
</tbody>
</table>
| Request body example | Request body example for a one-phase deployment that either uses the placement that is determined by the system, or does not use placement:  
```
{
  "deployment_name": "My Virtual System",
  "environment_profile_id": "1",
  "ssh_keys": ["ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQ...",
}
```
|                                     | Request body for the first phase of a two-phase deployment. Set "placement_only": true in the request body:  
```
{
  "deployment_name": "Two nodes_testname",
  "environment_profile_id": "1",
  "placement_only": true
}
```

Chapter 10. Reference 803
<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response headers</td>
<td>application/json</td>
<td></td>
</tr>
<tr>
<td>content-type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Table 180. Deploy a virtual system pattern (continued)**

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response body example</td>
<td>Response body example for a one-phase deployment that either uses the placement that is determined by the system, or does not use placement:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;status&quot;: &quot;RUNNING&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;deployment_id&quot;: &quot;d-79566e04e-0fac-49f2-b04e-efbc131a4c84&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;deployment_name&quot;: &quot;db2&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;app_type&quot;: &quot;application&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;app_id&quot;: &quot;a-3761fe57-2bda-4f9b-b90c-d2c435d69cb7&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;start_time&quot;: &quot;2011-03-25T17:02:57.878Z&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;virtual_system&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;id&quot;: &quot;1&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;instances&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;status&quot;: &quot;RUNNING&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;master&quot;: true,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;last_update&quot;: &quot;2011-03-25T17:11:13.750Z&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;private_ip&quot;: &quot;1xx.102.165.49&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;reboot.count&quot;: 0,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;stopped.by&quot;: &quot;&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;volumes&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>],</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;start_time&quot;: &quot;2011-03-25T17:03:51.654Z&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;id&quot;: &quot;rack9.xdbdev2b04.22b04.03473&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;database-db2.11301072577884&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;roles&quot;: [</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;node&quot;: &quot;database-db2.11301072577884&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;status&quot;: &quot;RUNNING&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;last_update&quot;: &quot;2011-03-25T17:11:14.840Z&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;external_uri&quot;: &quot;jdbc:db2://1xx.102.165.49:50000/mydb;user=dbpdbl;password=FgxmZv47TM8GwJD62Y1;&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;id&quot;: &quot;database-db2.11301072577884.DB2&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>],</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;public_ip&quot;: &quot;1xx.1xx.165.49&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;role_error&quot;: false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>],</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;role_error&quot;: false</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

Response body example for the first phase of a two-phase deployment. When "placement_only": true is included in the request body, placement is returned in the response body:

```javascript
{
  "placement": {
    "...
  },
  "deployment_url": "https://1xx.0.0.1:9443/services/deployments/d-55e08a31-a3f0-419e-b262-3c8ba2c6be2",
  "app_type": null,
  "topology": [
    {
      "component_id": "OS Node",
      "name": "OS Node",
      "parameters": [{
        "placement": true,
        "id": "WAS.PASSWORD",
        "label": "ADMIN_USER_PWD_LABEL",
        "description": "ADMIN_USER_PWD_DESCRIPTION",
        "type": "string",
        "displayType": "password"
      }],
      "scaling": {
        "min": 1,
        "max": 10,
        "init": 2
      }
    }
  }
}
```
Second phase of a two-phase deployment:

**PUT /resources/virtualSystemPatterns/**pattern ID

You can specify three parameters in the request body:

**placement**  
Required. The modified placement for the deployment.

**topology_parameters**  
Optional. Define the topology parameters that are needed for deployment as a key value map. The key format is topologyname.parameterid. For example, WAS.PASSWORD.

**addon_parameters**  
Optional. Define the topology parameters that are needed for deployment as a key value map.

Table 181. Deploy a virtual system pattern - second phase with modified placement

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URL</td>
<td><a href="https://localhost/resources/virtualSystemPatterns/a-cdaac959-672c-4df7-a648-b333a3843422">https://localhost/resources/virtualSystemPatterns/a-cdaac959-672c-4df7-a648-b333a3843422</a></td>
<td></td>
</tr>
<tr>
<td>Request content-type</td>
<td>application/json</td>
<td>Specify the updated placement in the request body.</td>
</tr>
</tbody>
</table>
Table 181. Deploy a virtual system pattern - second phase with modified placement (continued)

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request example</td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;operation&quot;: &quot;deployPlacement&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;topology_parameters&quot;: {},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;addon_parameters&quot;: {},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;placement&quot;: {</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;vm-templates&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;locations&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;1721665121&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;cloud_groups&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;esxset15&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;instances&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;new_instances&quot;: 1,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;nics&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;ip_groups&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;172&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;new_instances&quot;: 1,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;purpose&quot;: &quot;data&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;management&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;purpose&quot;: &quot;data&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;new_instances&quot;: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;1721665123&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;cloud_groups&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;esxset16&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;instances&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;new_instances&quot;: 1,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;nics&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;ip_groups&quot;: [{</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;172_2&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;purpose&quot;: &quot;data&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;new_instances&quot;: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;management&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;purpose&quot;: &quot;data&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;messages&quot;: [&quot;CWZKS6401E: 1721665123 is missing image: IBM OS Image for Red Hat Linux Systems:2.1.0.0.&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;new_instances&quot;: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>},</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;environment_profile&quot;: &quot;MyTest&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;name&quot;: &quot;Web_Application-was&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;new_instances&quot;: 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;version&quot;: &quot;5.0.0&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>
Table 181. Deploy a virtual system pattern - second phase with modified placement (continued)

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response code</td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td></td>
<td>404</td>
<td>The application that is specified by {appID} is not found.</td>
</tr>
<tr>
<td></td>
<td>412</td>
<td>A specified parameter is not valid. For example, the JSON file is not valid.</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

Update a virtual system pattern

PUT /resources/virtualSystemPatterns/pattern ID

Table 182. Update a virtual system pattern

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URL</td>
<td><a href="https://localhost/resources/virtualSystemPatterns/a-cdaac959-672c-4df7-a648-b333a3843422">https://localhost/resources/virtualSystemPatterns/a-cdaac959-672c-4df7-a648-b333a3843422</a></td>
<td></td>
</tr>
<tr>
<td>Request content-type</td>
<td>application/json</td>
<td></td>
</tr>
</tbody>
</table>
| Request example      | {
|                     | "content_type": "application/json",
|                     | "last_modifier": "tester",
|                     | "create_time": "2011-02-24T05:41:34Z",
|                     | "last_modified": "2011-02-24T05:41:34Z",
|                     | "access_rights": {
|                     | "tester": "F",
|                     | },
|                     | "content_md5": "5B8F7E6CF56F7CE80478C0086589AFF",
|                     | "app_type": "application",
|                     | "app_id": "a-fb70796e-1b13-467a-babe-b0b700bd563b",
|                     | "name": "App for Testing",
|                     | "locked": "false",
|                     | "creator": "tester"
|                     | } |
### Table 182. Update a virtual system pattern (continued)

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response code</td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td></td>
<td>404</td>
<td>The application that is specified by (appID) is not found.</td>
</tr>
<tr>
<td></td>
<td>412</td>
<td>A specified parameter is not valid. For example, the JSON file is not valid.</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>

### Delete a specified virtual system pattern

DELETE /resources/virtualSystemPatterns/{vsys_id}

### Table 183. Delete a specified virtual system pattern

<table>
<thead>
<tr>
<th>REST API information</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example URL</td>
<td><a href="https://localhost/resources/virtualSystemPatterns/a-cdaac959-672c-4df7-a648-b333a3843422">https://localhost/resources/virtualSystemPatterns/a-cdaac959-672c-4df7-a648-b333a3843422</a></td>
<td></td>
</tr>
<tr>
<td>Response content-type</td>
<td>application/json</td>
<td></td>
</tr>
<tr>
<td>Response code</td>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>401</td>
<td>The user is not authorized to perform this action.</td>
</tr>
<tr>
<td></td>
<td>403</td>
<td>Access forbidden</td>
</tr>
<tr>
<td></td>
<td>409</td>
<td>Conflict</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>Unexpected error</td>
</tr>
</tbody>
</table>
Chapter 11. Troubleshooting

Troubleshooting tools have been collected for ease of use when attempting to debug an issue.

**Before you begin**

You must be assigned the admin role to perform these steps.

**About this task**

The steps to troubleshoot an issue are different for each problem. To help make relevant information available to you as quickly as possible, the log files and other tools for troubleshooting problems have been consolidated together for convenience.

**Finding the log files**

To troubleshoot the IBM Cloud Orchestrator components, see the following table to find where the log files are stored.

<table>
<thead>
<tr>
<th>Component</th>
<th>Log file default path</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-service user interface</td>
<td>/var/log/scoui.log</td>
<td>Central Server 2</td>
</tr>
<tr>
<td></td>
<td>/var/log/scoui.trc</td>
<td></td>
</tr>
<tr>
<td>Administration user interface</td>
<td>/var/log/httpd</td>
<td>Central Server 2</td>
</tr>
<tr>
<td>OpenStack</td>
<td>/var/log/nova</td>
<td>Region Server</td>
</tr>
<tr>
<td></td>
<td>/var/log/glance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/var/log/cinder</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/var/log/heat</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/var/log/keystone</td>
<td>Central Server 2</td>
</tr>
<tr>
<td></td>
<td>/var/log/ceilometer</td>
<td>Central Server 1 and Region Server</td>
</tr>
<tr>
<td></td>
<td>/var/log/qpid</td>
<td></td>
</tr>
<tr>
<td>DB2</td>
<td>Collect the logs by running the following command: su -db2inst1 -s db2support</td>
<td>Central Server 1</td>
</tr>
<tr>
<td>Installer</td>
<td>/var/log/cloud-deployer</td>
<td>Deployment Server</td>
</tr>
<tr>
<td>Deployment Service</td>
<td>/var/log/ds</td>
<td>Deployment Server</td>
</tr>
<tr>
<td>HTTP Server</td>
<td>/var/log/httpd</td>
<td>Central Server 2</td>
</tr>
<tr>
<td>System Automation Application Manager</td>
<td>Collect the logs by running the following command in one line: /opt/IBM/tsamp/eez/bin/getamdata -all -outdir /tmp; zip saam_logs.zip /tmp/<em>-tsaam_data</em></td>
<td>System Automation Application Manager Server</td>
</tr>
<tr>
<td>haproxy</td>
<td>/var/log/haproxy.log</td>
<td>Central Server 2 and Region Server</td>
</tr>
<tr>
<td>Tivoli System Automation</td>
<td>/var.ibm/tivoli/common/eez/logs</td>
<td>Central Server 2 and Region Server</td>
</tr>
</tbody>
</table>
Table 184. Log files (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Log file default path</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Cloud Orchestrator</td>
<td>/var/log/ico_monitoring</td>
<td>Central Server 1 and Region Server</td>
</tr>
<tr>
<td>Monitoring</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public Cloud Gateway</td>
<td>/var/log/pcg.log</td>
<td>Central Server 2</td>
</tr>
<tr>
<td>Business Process Manager</td>
<td>/opt/ibm/BPM/v8.5/profiles/NodeProfile/logs</td>
<td>Central Server 2</td>
</tr>
<tr>
<td>Business Process Manager</td>
<td>/var/ibm/InstallationManager/logs</td>
<td>Central Server 2</td>
</tr>
<tr>
<td>installation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workload Deployer</td>
<td>/drouter/ramdisk2/mnt/raid-volume/raid0/logs/error/*</td>
<td>Central Server 3</td>
</tr>
<tr>
<td>Workload Deployer inlet</td>
<td>/drouter/ramdisk2/mnt/raid-volume/raid0/logs/trace/*</td>
<td>Central Server 3</td>
</tr>
<tr>
<td>Workload Deployer file server</td>
<td>/drouter/ramdisk2/mnt/raid-volume/raid0/usr/servers/fileserver/*</td>
<td>Central Server 3</td>
</tr>
<tr>
<td>Workload Deployer kernel</td>
<td>/drouter/ramdisk2/mnt/raid-volume/raid0/usr/servers/kernelservices/logs/*</td>
<td>Central Server 3</td>
</tr>
<tr>
<td>services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workload Deployer storehouse</td>
<td>/drouter/ramdisk2/mnt/raid-volume/raid0/usr/servers/storehouse/logs/*</td>
<td>Central Server 3</td>
</tr>
</tbody>
</table>

Problem determination with pdcollect tool

The script pdcollect.py is a log collection and log zip utility, which collects logs from all components in IBM Cloud Orchestrator.

The pdcollect.py script is located in /opt/ibm/orchestrator/pdcollect on the Deployment Server and Central Server 1.

You can start the script, after an error occurs, to collect as much information as possible and resolve the error. The script can zip remote files, directories, and logs from different hosts. Optionally, the script executes defined commands on the remote system and collects the output of the command execution in a zip file. The program can also execute batch scripts and collect the output. As a result, one convenient file is created that contains all logs and output of all hosts and components that are defined in the environment.

The pdcollect.py script uses two files, which must be in the same directory as the script itself: Environment.xml and Components.xml.

The Environment.xml file describes default installation host names and the components. The script uses this file as input to create the work Environment_work.xml file, which contains the additional dynamic compute nodes as reported by nova-manage service list, a command in OpenStack that lists available services.

The file Components.xml contains the relationships between the components and the actions that can be taken to collect the data and execute the commands on the specific systems. You can use the example Components.xml to look up the specific action that can be taken to gather output.
The dynamic list of compute nodes distinguishes between KVM and VMware nodes. The components that are assumed on the additional compute nodes need to have a specific name:

- computeNode: KVM compute host
- vmNode: VMware vCenter host
- esxNode: ESX hypervisor host

**Assumptions for using pdcollect tool**

- The script is executed as root on the Central Server 1.
- Hosts (remote systems) in Environment.xml are reachable from Central Server 1 through ssh.
- Additional compute nodes are gathered by nova-manage service list.
- Components of additional compute nodes are hardcoded.
- The script is executed as root on the Deployment Server.

**Running the pdcollect tool**

Run the `pdcollect.py` script on the Central Server 1 to collect logs from all components in IBM Cloud Orchestrator.

**Before you begin**

The `pdcollect.py` script is located in `/opt/ibm/orchestrator/pdcollect` on the Central Server 1.

The script requires the exchange of ssh keys to remote machines before execution.

Run the `pdcollect.py` script on the Deployment Server to collect Deployment service logs.

**Procedure**

On Central Server 1 or the Deployment Server, open the command line and run the following script as root:

```
python pdcollect.py [options]
```

where `[options]` are:

- `-h, --help`
  Shows this help message and exits.
- `-c Components.xml, --componentfile=Components.xml`
  Defines the input properties file name. Default name is `Components.xml`.
- `-v Environment.xml, --environmentfile=Environment.xml`
  Defines the environment file name. Default name is `Environment.xml`.
- `-o PDCollectlog, --output=PDCollectlog`
  Defines the output log file name. Default name is `PDCollectlog_<date and time in ISO format>`.zip.
- `-n SYSTEMLIST, --hostips=SYSTEMLIST`
  Defines the list of the host IP addresses defined in the environment file to be scanned for log files. Default value is to scan all hosts. Format is: `hostip1,hostip2,hostip3,...`
-p COMPONENTLIST, --components=COMPONENTLIST
   Lists the components to be scanned for the log files. The COMPONENTLIST format is component1,component2,component3,...

-s STARTDATE, --start=STARTDATE
   Defines the first date of the log sequence. The STARTDATE format is YYYY-MM-DD.

-e ENDDATE, --end=ENDDATE
   Defines the day after the last day of the log sequence. The ENDDATE format is YYYY-MM-DD.

--version
   Shows the pdcollect tool version and exits.

A disclaimer is displayed and logged first to alert you that the data that is gathered and stored may be confidential and could contain passwords.

Results

As output, a zip file container with the following name is created:
PDCollectlog_<date and time>_<hostname>.zip

Running the pdcollect tool with non-root user

You can run the pdcollect tool with non-root user.

Before you begin

The following commands are all run as root on the server indicated.

Procedure

1. Create a new user SSH for all the Central Servers and Region Servers:
   a. On each of the Central Servers and Region Servers:
      • Create a new user <yourmechid> and set the password:
        useradd -m <yourmechid>
pwd <yourmechid> #enter at the prompt <yourmechapwd>
      • Create the .ssh directory and set file permissions:
        su - <yourmechid> -c "mkdir .ssh; chmod 700 .ssh"
   b. On Central Server 1:
      • Generate the SSH keys for <yourmechid> and cp it to all IBM Cloud Orchestrator Servers:
        su - <yourmechid> -c "ssh-keygen -q -t rsa -N '' -f ~/.ssh/id_rsa"
      • Here $i stands for the IP address of each IBM Cloud Orchestrator server including Central Server 1:
        [root@cs-1] su <yourmechid>
        [yourmechid@cs-1] scp ~/.ssh/id_rsa.pub $i:~/.ssh/authorized_keys
        Note: Make sure that you accept the server key and the password required in <yourmechid>.
   c. Verify that <yourmechid> on Central Server 1 can SSH to all the IBM Cloud Orchestrator Servers including Central Server 1 without interruption:
      su - <yourmechid> -c "ssh <yourmechid>@$SCO_server_ip"

2. Copy the /opt/ibm/orchestrator/pdcollect and change the directory permission:
a. On Central Server 1, copy the directory /opt/ibm/orchestrator/pdcollect to /home/<yourmechid>:
   cp -rf /opt/ibm/orchestrator/pdcollect /home/<yourmechid>

b. Replace Components.xml with Components_nonroot.xml:
   cp Components.xml Components.xml.org
   cp Components_nonroot.xml Components.xml

c. Replace pdcollect.py with pdcollect_nonroot.py:
   cp pdcollect.py pdcollect.py.org
   cp pdcollect_nonroot.py pdcollect

d. Change the owner of /home/<yourmechid>:
   chown -R <yourmechid>:<yourmechidgroup> /home/<yourmechid>/

e. Modify the file pdcollect.py, and replace "yourmechid" with the new user name:
   # User which is used to execute remote commands
   SSH_USER = "yourmechid"

3. On each of the IBM Cloud Orchestrator servers, add the user <yourmechid> in the sudo list:
   a. Create a sudoer file named <yourmechid> and place it in /etc/sudoers.d:
      The content of the file <yourmechid> is as follows:

      Note: Replace <yourmechid> with your new user name.
      # sudoers additional file for /etc/sudoers.d/
      # IMPORTANT: This file must have no ~ or . in its name and file permissions
      # must be set to 440!!!
      # this file is for the SAAM mech-ID to call the SCO control scripts
      Defaults:<yourmechid> !requiretty
      # scripts found in control script directory
      # adapt the directory names to the mech id!
      # allow for
      <yourmechid> ALL = (root) NOPASSWD:/bin/su - db2inst1 -c db2support, (root) \ 
      NOPASSWD:/bin/find, (root) NOPASSWD:/bin/su, (root) NOPASSWD:/usr/bin/tee, (root) \ 
      NOPASSWD:/bin/netstat, (root) NOPASSWD:/bin/chmod, (root) NOPASSWD:/bin/rm, (root) \ 
      NOPASSWD:/usr/bin/zip, (root) NOPASSWD:/bin/cp

      b. Change the sudoer file permission:
         chmod 440 <yourmechid>

4. Run pdcollect.py with non-root user:
   See "Running the pdcollect tool" on page 813 to run pdcollect.py in /home/<yourmechid>/pdcollect/

   Note: While running ./pdcollect.py, it is fine to have an output message like "scp: XXXXXXXXXXXXXXXX: Permission denied". You can ignore these messages.

Setting logging levels

Set the logging levels of the IBM Cloud Orchestrator components to increase or decrease the collected troubleshooting information.

When the log data from IBM Cloud Orchestrator components does not provide enough details needed to determine the root cause of an error, many of the components have a configurable logging detail setting that you can increase to a debug level. Note that for some components, this should only be done on a temporary basis because log file sizes might increase dramatically when these components are configured to log in debug mode, and the file systems might run out of space.
To increase the logging level for the Self-service user interface, edit the 
/opt/ibm/ccs/scui/etc/log4j.properties file on Central Server 2, and replace 
all occurrences of INFO with TRACE. Restart the user interface with the following 
command:

```
service scui restart
```

To increase the logging level for the OpenStack Nova components, edit the 
/etc/nova/nova.conf on the Region Servers and add the following line in the 
[DEFAULT] section:

```
default_log_levels=amqplib=WARN,sqlalchemy=WARN,boto=WARN,suds=INFO,keystone=INFO,eventlet.wsgi.server=WARN, 
smartcloud=DEBUG,nova=DEBUG
```

You can change the logging setting for individual Nova components to WARN, 
INFO or DEBUG. After changing the logging level, you must restart the Nova 
components. For information about starting IBM Cloud Orchestrator services, see 
“Managing the services” on page 175.

To increase the logging level for the OpenStack Glance component edit the 
/etc/glance/glance*.conf files on the Region Server and change the Debug value to True. After changing the logging level, you must restart glance. For 
information about starting IBM Cloud Orchestrator services, see “Managing the 
services” on page 175.

To increase the logging level for the OpenStack Keystone component, edit the 
/etc/keystone/keystone.conf file on Central Server 2 and change level=WARNING 
to level=DEBUG in the [logger_keystone] section. After changing the logging 
level, you must restart the Keystone component by running the service 
openstack-keystone restart command.

To set the logging level for the Workload Deployer component, see “Workload 
Deployer log files” on page 817.

The Public Cloud Gateway component uses log4j logging. The 
log4j.properties file is located in the /opt/ibm/pcg/etc directory. For more 
information about the properties in the log4j.properties file, see the 

To set the logging level for the Business Process Manager component, log on to 
the WebSphere Integrated Solutions Console on the Central Server 2 and click 
Troubleshooting > Logs and trace to access the Logging and Tracing panel.

Log file rotation

IBM Cloud Orchestrator uses the logrotate mechanism of Linux to manage log file 
size and rotation settings. For information about adjusting the settings of the log 
files rotation, see the logrotate man page by running the man logrotate 
command. Note that you may be required to run the logrotate command using 
the -f flag after adjusting settings in the configuration files.

The OpenStack Nova, Cinder, and Glance log rotation settings are defined in the 
/etc/logrotate.d/openstack-* files on the Region Server.

The Keystone log rotation settings are defined in the /etc/logrotate.d/openstack- 
keystone files on Central Server 2.
Workload Deployer log files

The log files associated with Workload Deployer are stored on the machine where this component is installed (Central Server 3). The viewable logs can be viewed by using the user interface or they can be downloaded to your local file system for review.

Before you begin

You must be assigned the admin role to perform these steps.

About this task

There are 2 log files that can be viewed or downloaded, the error.log file and the trace.log file. Each log file has a maximum size that it cannot exceed and a file limit defining the number of versions of that file that can be maintained by Workload Deployer. Each error.log file can be a maximum of 2 MB in size and up to five versions of the file are maintained for a total of 10 MB of available data. Each trace.log file can be a maximum size of 100 MB in size and up to 10 versions of the file are maintained for a total of 1 GB of available data. If a version of either log file is older than 30 days, then it is automatically removed. Additionally, the oldest log file is removed if the file limit has been reached and a new file is created.

Procedure

1. Navigate to PATTERNS > Deployer Administration > Troubleshooting and expand the Logging section. By expanding the Logging section, you have access to the available logs using the log viewer and also be able to download the available logs to your file system for additional review.
2. Click one of the following links to open a new web page and view the log files:
   - View current error file to view the error log.
   - View current trace file to view trace log.
   a. Click Pause to stop new log entries from being appended. This action is only available if the log viewer is accepting new entries.
   b. Click Restart for new entries to be appended. This action is only available if the log viewer is not accepting new entries because it is paused.
   c. Click Clear to clear all the data from the log viewer. This action is available whether the log viewer is accepting new entries or if it is not accepting new entries.
3. Click Download log files to save all the available logs to your file system in .zip format. If you need to view information regarding events that have already happened, then you must use this link. A window is presented allowing you to open the compressed file or save it to your file system. The compressed file includes the current error.log file and the current trace.log file in their entirety, and the available archived versions of these logs. You can download all files with a single click.
4. Expand Configure trace levels to view or modify the trace levels. A set of default classes are defined as the trace string to be included in the logs. The level of trace for these classes can be modified and new classes can be added. The trace levels provided are based on Java Logging convention and WebSphere Application Server levels. The complete list of trace levels is listed later in this section, ordered in ascending order of severity:
FINE: The trace information is a general trace plus method entry / exit / return values.

FINER: The trace information is a detailed trace.

FINEST: the trace information is an even more detailed trace that includes all the detail that is needed to debug problems.

ALL: All events are logged. If you create custom levels, ALL includes your custom levels and can provide a more detailed trace than FINEST.

SEVERE: The task cannot continue, but the component can still function.

WARNING: Potential or impending error.

INFO: General information outlining the overall task progress.

OFF: No events are logged.

Increasing logging will decrease performance. You might need advice from IBM Customer Support if you want to change the trace levels.

a. Add a trace string. Click Add trace string and enter in a valid trace string. The trace level for a new trace string is set to INFO by default.

b. Remove a trace string. Click the remove icon next to a trace string to remove that trace string.

c. Modify a trace level. Click the <trace_string> and select a new trace level in the drop down menu. Click Save to commit the new trace level for the specified trace string.

Results

After you have completed these steps, you have reviewed all the available log data.

Related reference:

- Problem determination command-line interface reference

You can work with IBM Cloud Orchestrator problem determination using the command-line interface.

“Example script to configure the trace levels” on page 382

This script package sets a trace specification level (example "com.ibm.ws.ibm.*=info") on all servers in a cell. It can be included on either a stand-alone pattern part or a Deployment Manager pattern part. Users can specify the trace specification during deployment.

Known errors and limitations

Check the following sections for information about known errors and limitations in IBM Cloud Orchestrator.

Product limitations

Review the following list of limitations of IBM Cloud Orchestrator.

- The Power NPIV feature requires that all of the hosts in a given system pool have NPIV-capable Fibre Channel adapters.

- Network adapter of type E1000E is not supported by IBM Cloud Orchestrator. You cannot deploy images containing this type of network adapter.

- For OpenStack, the service users (nova, cinderglance, heat, ceilometer ...) must not be renamed and must be enabled. Also the service project must not be renamed and remain enabled. IBM Cloud Orchestrator adds to that: the administrator user as well as project admin (admin tenant) must not be renamed or disabled.
For z/VM and deploy of an image to an ECKD™ disk which is larger than the source ECKD disk or a resize via admin UI to a flavor with larger disk, the additional space will not be usable until the user re-partitions the disk and re-sizes the file system. Using a flavor with disk size of 0 will cause the virtual machine to be created with the same disk size as the source disk and therefore avoid the issue. This issue occurs only with ECKD disks. For Power and VMware Regions, some of the image's configurations will affect instances' flavor, for virtual system instances deployed from them. Their flavor may be renamed to a format like Instance_UUID when using the `nova show <instance_id>`. You can get the correct instance flavor from IBM Cloud Orchestrator using the following steps:

1. Log in to IBM Cloud Orchestrator.
2. Click PATTERNS > Instances > Virtual System Instances.
3. Select YOUR_INSTANCE_NAME and in the Virtual machines section, expand your virtual machine and see the Flavor in the Hardware and network section.

Inject SSH keys to virtual machine is not supported in VMware region. When you perform the following OpenStack procedure, the SSH keys is not injected:

1. Generate the SSH keys with `ssh-keygen`.
2. Add a public key to OpenStack with `nova` command or Administration user interface:
   
   ```bash
   nova keypair-add --pub_key id_rsa.pub <keyname>
   ```
3. Deploy the machine using the key provided by OpenStack:
   
   ```bash
   nova boot --image <imageID> --key_name <keyname> --flavor 2 <vmName>
   ```
4. You can use this key to access the virtual machine:
   
   ```bash
   ssh -i <your private key> user@vm_ip_address
   ```

**Unable to reduce the disk size of VMware and KVM virtual machines**

Changing the flavor does not reduce the disk size of VMware and KVM virtual machines.

OpenStack provides the API and CLI to resize an existing virtual machine to a different flavor. You can change the flavor to scale the disk size: up or down. The original virtual machine is saved for a period of time so that the change can be rolled back if a problem occurs. Test and explicitly confirm all resizing. When you confirm a resize, the original virtual machine is removed. If you do not explicitly confirm or revert the changes, all resizes are automatically confirmed after 24 hours.

Some hypervisors do not support scaling down the disk size:

**VMware**

You cannot resize the virtual machine from a large disk size to a smaller disk size. The resize operation fails with an error message.

**KVM**

No error message is displayed, but the disk size is not reduced.
**Hypervisor errors**

You can receive error messages for hypervisors defined to IBM Cloud Orchestrator under certain circumstances.

**Minus (-) free disk is displayed in OpenStack for command nova hypervisor-show**

**Problem**

After the deployment of IBM Cloud Orchestrator, use the following commands to get the nova information:

```
[root@vmware-region-pattern1 images]# nova hypervisor-show 1
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>1</td>
</tr>
<tr>
<td>Hypervisor hostname</td>
<td>computenodeB</td>
</tr>
<tr>
<td>Property</td>
<td>Value</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cpu_info_model</td>
<td>[&quot;Intel(R) Xeon(R) CPU X5560 @ 2.80GHz&quot;, &quot;Intel(R) Xeon(R) CPU X5570 @ 2.93GHz&quot;]</td>
</tr>
<tr>
<td>cpu_info_topology_cores</td>
<td>16</td>
</tr>
<tr>
<td>cpu_info_topology_threads</td>
<td>32</td>
</tr>
<tr>
<td>cpu_info_vendor</td>
<td>[&quot;IBM&quot;, &quot;IBM&quot;]</td>
</tr>
<tr>
<td>current_workload</td>
<td>0</td>
</tr>
<tr>
<td>disk_available_least</td>
<td>-</td>
</tr>
<tr>
<td>free_disk_gb</td>
<td>25</td>
</tr>
<tr>
<td>free_ram_mb</td>
<td>-43934</td>
</tr>
<tr>
<td>host_ip</td>
<td>172.16.144.60</td>
</tr>
<tr>
<td>hypervisor_hostname</td>
<td>domain-c7(HA-Cluster1)</td>
</tr>
<tr>
<td>hypervisor_type</td>
<td>VMware vCenter Server</td>
</tr>
<tr>
<td>hypervisor_version</td>
<td>5001000</td>
</tr>
<tr>
<td>id</td>
<td>1</td>
</tr>
<tr>
<td>local_gb</td>
<td>1919</td>
</tr>
<tr>
<td>local_gb_used</td>
<td>2171</td>
</tr>
<tr>
<td>memory_mb</td>
<td>89698</td>
</tr>
<tr>
<td>memory_mb_used</td>
<td>133632</td>
</tr>
<tr>
<td>running_vms</td>
<td>18</td>
</tr>
<tr>
<td>service_host</td>
<td>vmware-region-pattern1</td>
</tr>
<tr>
<td>service_id</td>
<td>6</td>
</tr>
<tr>
<td>vcpus</td>
<td>32</td>
</tr>
<tr>
<td>vcpus_used</td>
<td>58</td>
</tr>
</tbody>
</table>

**Solution**

`disk_available_least` for hypervisor can be a negative number to indicate the over commitment of hypervisor disk space.

The qcow2 disk format is used for the virtual machine in the KVM hypervisor, the whole size of disk will not be allocated from beginning to save the disk space, `disk_available_least` comes from the following equation:

\[
\text{disk\_available\_least} = \text{free\_disk\_gb} - \text{disk\_overcommit\_size}
\]

\[
\text{disk\_overcommit\_size} = \text{virtual size of disks of all instance instance - used disk size of all instances}
\]

When the hypervisor instances overcommitted more disk space than free disk space, `disk\_available\_least` is a negative number.

**Minus (-) free_ram_mb or current_workload is displayed in OpenStack for command nova hypervisor-show**

```
[root@vmware-region-pattern1 images]# nova hypervisor-show 1
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu_info_model</td>
<td>[&quot;Intel(R) Xeon(R) CPU X5560 @ 2.80GHz&quot;, &quot;Intel(R) Xeon(R) CPU X5570 @ 2.93GHz&quot;]</td>
<td></td>
</tr>
<tr>
<td>cpu_info_topology_cores</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>
The free_ram_mb for hypervisor can be a negative number to indicate the over commitment of hypervisor memory. The default memory overcommit rate is 1.5, that means you can use memory overall memory_mb * 1.5 memories. The default cpu overcommit rate is 16, that means you can use memory overall vcpus * 16 vcpus.

To configure the overcommit rate, you must modify the following attribute in nova.conf and restart the openstack-nova-scheduler and the openstack-nova-compute services.

Note: This configuration is effective for KVM, PowerVC, and VMware regions.

# virtual CPU to Physical CPU allocation ratio (default: 16.0)
cpu_allocation_ratio=16.0

# virtual ram to physical ram allocation ratio (default: 1.5)
ram_allocation_ratio=1.5

OpenStack fails to resize Kernel-based virtual machines

Problem
If you try to resize a Kernel-based virtual machine that uses the config_drive parameter, the status of the virtual machine changes to ERROR.

Solution
Do not resize a virtual machine that uses the config_drive parameter. You can use the following command to check it:
nova show instance_id/instance_name

Image errors

There are some known errors that might occur when managing images in IBM Cloud Orchestrator.

Error deploying a Windows virtual image

When deploying a Windows virtual image in a pattern, the following Microsoft error is displayed in the hypervisor console:

Windows could not parse or process the unattend answer file for pass [specialize].
The settings specified in the answer file cannot be applied.
The error was detected while processing settings for component [Microsoft-Windows-Shell-Setup].
Causes

This error might happen if you specified the wrong product key for the Windows operating system of the image you are deploying.

Resolving the problem

Configure the virtual image part in the pattern by specifying the right product key for the Windows operating system and deploy the virtual image again.

Limitation: When deploying a Windows image on VMware, you must match the number of NICs with the number of networks and IP addresses used for deployment

This limitation occurs when you deploy Windows images using VMware. For example, a Windows image with two defined NICs must be deployed using two or more networks. If you try to deploy it using only one network, the deployment will fail.

Cannot reach the virtual machine deployed in nova

In `/etc/sysconfig/network` comment or remove `GATEWAYDEV=<some-if>`.

This directive might cause the deployed virtual machine routing table to be set incorrectly and therefore the virtual machine might result unreachable.

Disabling VNC in `nova.conf` can cause some images to not display completely

If you disabled the VNC in `nova.conf` (`vnc_enable=false`) then some images might not deploy completely. You can see a message like `boot from harddisk` for the instances in the console log. The reason is that OpenStack does not prepare the graphic device for the instance if the VNC is disabled. Therefore, if the image is depending on the graphic during the boot time, it will hang and will not boot up.

Cannot complete the deployment of an image

Deleting a flavor and creating a new one with the same ID, the old flavor is always used in Workload Deployer even when the user selects to use the new flavor.

This is a known OpenStack issue. The deleted flavor is always returned if the new one has the same ID. The workaround is to create a new flavor with a different ID.

Cannot import a virtual image

When importing a virtual image, the following error message is displayed: CWZCO1030E: An image with name `<image_name>` is already defined in the database.

but an image with the same name is not displayed in the image list.

Causes

The problem occurs if you are importing a virtual image with the `catalogeditor` role and another user already imported an image with the same name.
Resolving the problem

Ask a user with the admin role to perform one of the following actions:

- Granting the access to the existing virtual image to the project to which your user belongs.
- Registering the new virtual image with a different name and then granting the access to the virtual image to the project to which your user belongs.

Instance errors

There are some known errors that might occur when managing instances in IBM Cloud Orchestrator.

OpenStack reports an rpc error in the log when deleting an instance

The log might show:
```
TRACE nova.openstack.common.rpc.amqp
File "/usr/lib/python2.6/site-packages/nova/compute/manager.py", line 923,
in _delete_instance nova.openstack.common.rpc.amqp instance["uuid"]
TRACE nova.openstack.common.rpc.amqp
File "/usr/lib/python2.6/site-packages/nova/consoleauth/rpcapi.py", line 68,
in delete_tokens_for_instance
TRACE nova.openstack.common.rpc.amqp instance_uuid=instance_uuid)
```

Solution:

Install openstack-nova-console*.rpm that can be found in your IBM Cloud Orchestrator package and then run:
```
/etc/init.d/openstack-nova-consoleauth restart
```

Failed to deploy instance, it shows "No valid host was found"

When you deploy instance, it shows ERROR immediately and use nova show <instance name>, it shows No Valid host was found.

This is because the nova scheduler is not able to find a hypervisor to provision the virtual machine, this usually happens when:

1. Not clean unused virtual machine for a long time.
2. Provisioning more virtual machines than the cloud capacity.
3. Disable or remove hypervisor from cloud like remove ESXi host from VMware vCenter, KVM compute is down/offline.

To understand if the cloud is out of capacity, you can use nova hypervisor show <hypervisor id> to check CPU/memory/disk that is used by the virtual machines on each hypervisor.

To debug the problem, you can modify the /etc/nova/nova.conf and change the debug and verbose to true as following:
```
default=false
debug=true
verbose=true
```
Restart the openstack-nova-scheduler and after a failure, you should be able to see following messages in /var/log/nova/scheduler.log:

```
2014-09-16 10:15:37.672 28695 DEBUG nova.scheduler.filters.ram_filter [req-ef6b203c-e0fa-4ba0-8f9f-b2d71ca3deb9 e417380bc10b48c1b5fb4296d6fa470d 26e0b2a976784f88cd62d7580682bf0] (cil017110014, domain-c19(cluster110)) ram:-42019 disk:267230208 io_ops:0 instances:24 does not have 2048 MB usable ram. host_passes
/usr/lib/python2.6/site-packages/nova/scheduler/filters/ram_filter.py:60
2014-09-16 10:15:37.673 28695 INFO nova.filters [req-ef6b203c-e0fa-4ba0-8f9f-b2d71ca3deb9 e417380bc10b48c1b5fb4296d6fa470d 26e0b2a976784f88cd62d7580682bf0] Filter RamFilter returned 0 hosts
2014-09-16 10:15:37.673 28695 WARNING nova.scheduler.driver [req-ef6b203c-e0fa-4ba0-8f9f-b2d71ca3deb9 e417380bc10b48c1b5fb4296d6fa470d 26e0b2a976784f88cd62d7580682bf0] [instance: a4086e77-389e-4eb8-9431-de5eb6da97f0] Setting instance to ERROR state.
```

Which indicate the RAM is not enough to host one more virtual machine with 2 GB required.

## Deployment errors

There are some known errors that might occur when deploying virtual system patterns and images in IBM Cloud Orchestrator.

### Setting the host name when deploying a Windows system

When deploying a Windows system, you must specify a `computername`.

The `computername` value is set as the virtual instance host name. If you want to set the host name of the virtual instance to something different than the string specified in `computername`, you can create a script package and add it to the image in the virtual system pattern.

The content of the script can be similar to the following sample:

```
nslookup $IPADDR | findstr Name > %TEMP%\oldhostname
set /p str=%TEMPS%\oldhostname
echo."%str%"

set str=%str: =%
echo."%str%"
set str=%str:Name:=%
echo."%str%"
wmic COMPUTERSYSTEM where "Name='COMPNAME'" CALL Rename str, Password, User
reboot
```

where

- `IPADDR` is the value passed from IBM Cloud Orchestrator as `${partname.ipaddr}`
- `COMPNAME` is the value you specify in `computername`.
- `Password` is the administrator password.
- `User` is typically Administrator.

**Attention:** This script example reboots the image.

This limitation impacts the usage of property variables in virtual system patterns when dealing with Windows parts, because `${part-name_.hostname}` is not resolved correctly. The workaround in this case is to use the `${part-name_.ipaddr}` parameter instead. For more information about the property variables, see [Properties variable syntax](#).
Unable to deploy a virtual system pattern with a non-admin user

You might be unable to deploy a virtual system pattern or virtual application pattern with a non-admin user.

**Symptom**

The deployment fails with the following in the Workload Deployer log:

```java
com.ibm.ccs.openstack.shim.task.RegisterTask register About to create server:
SCO-172-16-30-0/24-Web_Application-was.1137926855084
com.ibm.ccs.openstack.shim.task.RegisterTask register Error creating OpenStack instance
com.ibm.openstack.api.exceptions.OpenStackItemNotFoundException: The resource could not be found.
at com.ibm.openstack.api.OpenStackException.<init>(OpenStackException.java:12)
at com.ibm.openstack.api.OpenStackBadResponseException.<init>(OpenStackBadResponseException.java:12)
at sun.reflect.GeneratedConstructorAccessor16.newInstance(Unknown Source)
```

**Solution**

To solve this problem, perform the following steps:

1. All networks created in OpenStack must be created with `--project <project id>` parameter specified. If a network is to be shared across multiple users, the easiest way is to define a Public project and include all users in that project.

   Because in a multi-tenancy scenario each project must have its own network created and assigned, make sure that your project has one network attached. For example, project003 has network 172.16.30.0/24 attached, then the member of project003 can deploy successfully. Verify that the specified network ID belongs to the project that you currently use, by using the following commands:

   ```bash
   [root@SVT-CIL-NEW ~]# keystone tenant-list
   +----------------------------------+------------+---------+
   | id | name | enabled |
   +----------------------------------+------------+---------+
   | 1f9f8b62052046ee97763f4eb88288e3 | service | true |
   | 3c8b192caab1499aa44eb2dcf420b9a12 | admin     | true |
   | c7ea7db95d2241c383f2f5995b31fa19   | project003| true |
   +----------------------------------+------------+---------+
   [root@SVT-CIL-NEW ~]# nova-manage network list
   id  IPv4    IPv6   start address   DNS1   DNS2    VlanID
   1   172.16.30.0/24 None   172.16.30.10  172.16.30.2 172.16.30.2 4090
   project ...
   c7ea7db95d2241c383f2f5995b31fa19 ...
   ```

2. If you want to associate an existing network to a project, run the following command:

   ```bash
   nova-manage network modify <x.x.x/yy> <project_id>
   ```

   where

   - `<x.x.x/yy>` is the network to be modified.
   - `<project_id>` is the ID of the project to be associated.

   For example:

   ```bash
   nova-manage network modify 172.16.30.0/24 c7ea7db95d2241c383f2f5995b31fa19
   ```
Error displayed when deploying a virtual system pattern
You might see the message "Missing cloud or ip group configuration on parts. Please make a selection before deployment."

Symptoms:
When deploying a pattern, for example, a virtual system pattern with a Linux image (RHEL-mini), an error message might be returned indicating that a cloud or group IP is missing: "Missing cloud or ip group configuration on parts. Please make a selection before deployment."

Causes:
This occurs because the cloud or IP group was not specified and, in some cases, the Flavor field might also be empty.

Solution:
After selecting Deploy, select Configure virtual parts and enter the required details.

Unable to deploy WebSphere Application Server OVA image
When you try to deploy the WebSphere Application Server OVA image on PowerVM or VMware, the operation might hang.

Solution
To resolve the problem, set the disk size of the image flavor to 0. The flavor then uses the default size of the image to be deployed.

Alternatively, set the disk size of the flavor to a value that is equal to, or greater than, the size of the image to be deployed. To identify the correct value, use the nova image-show image_ID | grep disk command, as shown in the following example:

```
nova image-show 4223c31a-4fcf-1747-b3e3-478d44510201 | grep disk
```

```
| metadata customization.disksize.hard disk 1
| {"category": "Storage Settings", "name": "DiskSize.Hard disk 1", "classification": {"id": "STORAGE", "label": "Storage"}, "rules":
| ["id": "increment", "value": "1"],
| {"id": "incrementType", "value": "LINEAR"},
| {"id": "max", "value": "23538"},
| {"id": "min", "value": "12288"}],
| "required": false, ["value": 12288],
| "type": "LONG",
| "description": "Disk Size of Hard disk 1 (MB)"]
```

Note: Line breaks and indents have been inserted in the command output, to make the example easier to read.
In this example, the disk size required to create an instance of this image is 12288.

Deployment of the virtual system pattern fails due to the name of the virtual machine
When deploying a virtual system pattern to an environment profile that has custom definition of the virtual machine name property using the ${hostname} variable, the deployment fails.

Symptoms
The deployment history ends with the following message:
Virtual machine could not be registered date/time
In addition, when browsing the Virtual Machines section of the failed instance, you see that the name of the virtual machines containing just the first octet of the IP address instead of the host name, for example d_172 when the IP address of the machine is like 172.16.*.*.

**Causes**

The DNS entry is missing in the forward lookup zone for the IP address which has been assigned to the virtual machine. You can find the exact address in the hypervisor tools (vCenter or OpenStack).

**Resolving the problem**

There are the following ways to solve this issue:

- Update the DNS configuration, so that it correctly resolves the host name for the given IP address. This is the preferred way.
- Change the virtual machine name format property from the environment profile definition to not use `${hostname}` or remove it completely.

**Script execution does not report failing condition**

When deploying a virtual machine on Power with an additional user created through add-on, the potential failures are not shown as a failed status of the script.

**Symptoms**

The user is not properly created on the newly provisioned Power virtual machine even if the add user script has run and is marked as successfully completed on the virtual machine details.

**Causes**

This is an issue with the add-on script being used, which is not properly handling the error condition and always completing with success.

**Resolving the problem**

You must perform additional manual steps to verify that user has been properly created:

- Open the Scripts section within the particular virtual machine in the pattern instance.
- Locate the user add-on script.
- Download the `remote_std_err.log` log and verify if it contains any error messages.
**No valid host was found**
Exceeded the maximum scheduling attempts of three for instance `<instance_id>`.

**Cause**
The actual deployment of a virtual machine failed since an error occurred while placing the virtual machine in the cloud. This error message hides the actual root cause of the problem.

**Resolving the problem**
To identify the root cause of the problem, the administrator must check the `nova.log` of the Compute Node in the corresponding region. The reason of the problem could be caused by a bandwidth of issues, like datastore is full, disk is out of space, not enough IP addresses available, and so forth.

**nova command errors and limitations**
Check errors and limitations when using the nova command.

**nova command limitation**
Nova client does not work with users outside of the default domain defined in OpenStack, so only users in default domain can run the `nova` command.

**The `nova-manage` command is not able to validate the project ID**
This happens if you update quotas or networks for a project that does not exist, and so the `nova-manage` command is not able to use the keystone API to verify if the project exists.

The OpenStack community decided to deprecate `nova-manage` in future releases for everything except the `db-sync` command, and move all functionality out of it into APIs. For details, you can see a blueprint at [https://blueprints.launchpad.net/nova/+spec/apis-for-nova-manage](https://blueprints.launchpad.net/nova/+spec/apis-for-nova-manage). Due to these limitations and the deprecation of the `nova-manage` command, this problem will not be fixed. In the current IBM Cloud Orchestrator release, the `nova-manage` command can still be used to manage nova resources.

**Security limitations**
Check the known security limitations that might expose your IBM Cloud Orchestrator environment to risks.

**Toolkit parameters are saved in the log file**
The SCOrchestrator toolkit saves in the log file all parameters passed by other toolkits used by runbooks. If these parameters contain security sensitive information, they are visible in the `<path to BPM profile>/log/bpm4scol/SystemOut.log` file.

Refer to “Troubleshooting Business Process Manager” on page 840 for detailed information on this security limitation.
Unable to change the flavor of VMware virtual machines

Attempts to change the flavor of VMware virtual machines by using OpenStack fail with an error.

If:
• You try to change the flavor of VMware virtual machines by using OpenStack, the OpenStack server object is in an error state.
• The virtual machine itself is not affected.

You must verify the following settings for /etc/nova/nova.conf:
• If there is only one compute node, set allow_resize_to_same_host to true.
• Set multi_host to false.

You must also make sure that the virtual machine uses an SCSI disk.

Error message displayed when trying to start a virtual system

The resource could not be found message might be displayed when a user from a newly created project tries to start a virtual system.

An example of this situation is when you run the following command to run a virtual machine from OpenStack:
nova boot --image rhelova --flavor m1.large --nic net-id=e325a701-ab07-4fb9-a7df-621e06b31c9b test1vm2

The following message is displayed:

ERROR: The resource could not be found. (HTTP 404) (Request-ID: req-4d801615-5fb3-49fe-8830-860de3f3f7db)

The reason for the error is that the user cannot access the network that is defined in the environment profile for the virtual system. This problem occurs when a user who is associated with one project (for example, Project A) tries to access a network that is associated with another project (for example, Project B).

The problem can be caused because of the following issues:
• The network was manually associated with Project B, as described in “Associating a Nova network to a project” on page 108.
• The network was automatically associated with Project B when an admin user from Project B booted a virtual machine from OpenStack without specifying the network:
  – Using the API: Omitted the networks parameter
  – Using the CLI: Omitted the --nic parameter

If the network is not specified in the boot command, and if the project of the current user is not associated with any network, OpenStack associates the network with the project of the current user: in our example, Project B.

If a network is associated with a project, that network cannot be accessed by users from any other project.

To resolve this problem, you must associate the network with the appropriate project to ensure that the users assigned to that project can access that network.

You can associate a network with only one project. If you want users from multiple projects to access the same network, you must disassociate all projects from the network, which sets the project_id value for the network to None.
For details about how to associate a network with a project, and how to disassociate all projects from a network, see "Associating a Nova network to a project" on page 108.

Unable to list keystone endpoints on the Region Server

You are unable to list keystone endpoints on the Region Server.

This limitation is due to OpenStack and the keystone version 3 support not being complete. To support keystone version 3, the keystone client must use OS_SERVICE_TOKEN and OS_SERVICE_ENDPOINT.

Symptoms:

The keystone endpoint URL is empty when running the keystone endpoint-list command on the Region Server with the /root/openrc sourced:

```
export openrc
keystone endpoint-list
```

<table>
<thead>
<tr>
<th>id</th>
<th>region</th>
<th>publicurl</th>
<th>internalurl</th>
<th>adminurl</th>
</tr>
</thead>
<tbody>
<tr>
<td>019e393701f5436480d6ec9552cd0642</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08f15c7f3b76fa4ada720944cdcc05</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1c76d4c3cd7e4eb683f22021dfbceee</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4084c21c7f4477b9d125d06b5b190</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8203260f667604b8114350781779d0</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>895206c5714dfcb8c0ca657b8e2b5</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>97781db45f6e4d69768220025f37523</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a8917ae2b383a96a31c3976f93a6f2</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a752e1491f4cb5b88f1833aa1c49</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b41db713b4ac1b145f4172f7231e4e</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b52a97b7c3af42e1bc6b7b8f10dc01</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c8d4aee1f4e42c84f5f3f14a5191a6</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d8d615e0a0745e88ad6f8d137998c</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>df591717295c4299cb36d27264c</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d6632f4e0144189a41e35385b6f555</td>
<td>RegionOne</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solution:

1. Copy /root/keystonerc from central-server-2. Change the OS_REGION_NAME variable to the corresponding region name.
2. Source the copied keystonerc:
source keystonec

3. Rerun the keystone endpoint-list command.

Unable to log in by using an LDAP user

You are unable to log in to IBM Cloud Orchestrator with an LDAP user after configuring Microsoft Active Directory.

Symptoms:
As an LDAP user, if you try to run any command in OpenStack, for example, `nova list`, you get the following error: `ERROR: Invalid OpenStack Nova credentials`.

Causes:
When searching from the domain level, Active Directory returns referrals (search continuations) for some objects, to indicate to the client where to look for these objects. Client-chasing of referrals is a broken concept, since LDAP v3 does not specify which credentials to use when chasing the referral. Windows clients are supposed to use their Windows credentials, but this does not work in general when chasing referrals received from and pointing to arbitrary LDAP servers. As a result, the default `libldap` automatically chases the referrals internally with an anonymous access, which fails with Active Directory.

Solution:
To resolve this issue, you must switch off this behavior. Edit `/usr/share/pyshared/keystone/common/ldap/core.py` to add a new line under the `ldap.initialize` line:

```
self.conn.set_option(ldap.OPT_REFERRALS,0)
```

Unable to add disk to SLES instance

SLES hotplug of virtual disk is not fully supported.

Symptoms
The Default add disk add-on failed or the device requested in the Default raw disk add-on is not present in the `fdisk -l` output.

Resolving the problem
Manually request a reboot from the virtual machine or stop and restart it from the user interface, and click Execute Now in the virtual machine script packages section.

Errors of 32-bit library files `libstdc++.so.6` and `/lib/libpam.so*` were not found in `db2prereqcheck.log`

The following errors are not critical and can be ignored.

```
Validating "32 bit version of "libstdc++.so.6" " ...
Found the 64 bit "/usr/lib64/libstdc++.so.6" in the following directory "/usr/lib64".
DBT3514W The db2prereqcheck utility failed to find the following 32-bit library file: "libstdc++.so.6".
Validating "/lib/libpam.so*" ...
DBT3514W The db2prereqcheck utility failed to find the following 32-bit library file: "/lib/libpam.so*".
WARNING : Requirement not matched.
Requirement not matched for DB2 database "Server" . Version: "10.5.0.2".
Summary of prerequisites that are not met on the current system:
DBT3514W The db2prereqcheck utility failed to find the following 32-bit library file: "/lib/libpam.so*".
```
Unable to reach one of the addresses if multiple NICs of a Linux virtual machine are deployed

The router of the second IP address is not set in the virtual machine automatically.

Problem

The problem occurs because in Linux there is only one default gateway, which means that even if the network packet can reach the second NIC, the response packet still uses the default gateway. At that point, the response packet is not able to reach the sender.

Solution

The solution is to manually add another routing table by performing the following steps:

1. Determine which is the default gateway and which NIC needs to add an additional route table. Run the command:

```
$ ip addr show
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
1: eth0 <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1454 qdisc pfifo_fast state UP qlen 1000
link/ether fa:16:3e:cd:c3:17 brd ff:ff:ff:ff:ff:ff
inet 172.16.201.145/24 brd 172.16.201.255 scope global eth0
inet6 fe80::f816:3eff:fe6c:317/64 scope link
valid_lft forever preferred_lft forever
```

Now, the virtual machine has two NICs: eth0 has 172.16.201.145, eth1 has 172.16.202.4.

2. You must add another route table for eth0. Use following command (eth0 is the name of the route table or you can provide your own meaningful name):

```
echo "1 eth0" >> /etc/iproute2/rt_tables
```

3. Configure the routing rules for table eth0:
Unable to access the Administration user interface
You are unable to access the Administration user interface.

Problem
A user cannot access the Administration user interface after successful installation. This is not a common error and occurs only if the node runs out of space.

Solution
Check if there is enough space on Central Server 2 in and if the user has created any softlink to manage the space on the partition. Then, make sure that the targeted folder has write permissions. The installer requires write permission on directories, such as /tmp, /var/tmp, /usr/tmp, /var/www. If you have mounted these directories on the target directory, make sure that they have write permission by group, for example 777 permission on /tmp, /var/tmp.

User name and password for vCenter are incorrect
At the beginning of the installation of a VMware Region, you are required to input information for the vCenter. If you enter an incorrect user name or password for vCenter, the VMware Region does not connect to the vCenter.

Problem Determination
If you get nothing when running `nova list` and `nova hypervisor-list`, check the following files and services:

1. `/var/log/nova/compute.log`:
   
   2014-08-05 23:40:38.779 10710 ERROR suds.client [-] <xml version="1.0" encoding="UTF-8"?>
   <ns1:Body>
   <ns0:Login>
   <ns0:userName>root</ns0:userName>
   <ns0:password>object00!</ns0:password>
   </ns0:Login>
   </ns1:Body>
   </soap-ENV:Envelope>
   2014-08-05 23:40:38.782 10710 CRITICAL nova.virt.vmwareapi.driver [-] Unable to connect to server at 172.19.4.9, sleeping for 60 seconds
   2014-08-05 23:40:38.782 10710 TRACE nova.virt.vmwareapi.driver Traceback (most recent call last):
   2014-08-05 23:40:38.782 10710 TRACE nova.virt.vmwareapi.driver File "/usr/lib/python2.6/site-packages/nova/virt/vmwareapi/driver.py", line 1067, in <module>
   2014-08-05 23:40:38.782 10710 TRACE nova.virt.vmwareapi.driver password=self._host_password)

2. `/etc/init.d/openstack-nova-compute status`:

   [root@SCO24-a28-node2 init.d]# ./openstack-nova-compute status
   openstack-nova-compute (pid 10710) is running...

Note: The service openstack-nova-compute is supposed to be running even if the user name or password is wrong. This service cannot connect to the vCenter but by itself is OK.
Validate user name and password

Provide the correct user name and password. You can ask for it from your vCenter administrator. Then you can validate the user name and password in the following ways:
1. Use vsphere to log into the vCenter with user name and password.
2. Use webclient to log into the vCenter with user name and password:
   

If the user name and password are correct, the login is successful.

Configure the /etc/nova/nova.conf

Use the following commands to configure /etc/nova/nova.conf:

   openstack-config --set /etc/nova/nova.conf vmware host_username vCenter_username
   openstack-config --set /etc/nova/nova.conf vmware host_password vCenter_password

Note: If you want the vCenter_password encrypted, you can use
   openstack-obfuscate vCenter_password first.

Restart openstack-nova-compute

You must restart openstack-nova-compute:

   service openstack-nova-compute restart

Hypervisors remain in maintenance mode

After Workload Deployer deployment, the hypervisors remain in maintenance mode. To start them again, perform the following procedure.

Solution

1. Log on to the Self-service user interface as an administrator.
2. Go to PATTERNS > Deployer Configuration > Cloud Groups.
3. For each cloud group, expand the cloud hardware section and check the status of the related hypervisors.
4. For each hypervisor in maintenance mode, click the hypervisor name to open the Hypervisor window.
5. Expand the Networks and Storage devices sections and ensure that there is at least one network and storage device marked as in use.
6. Start the hypervisor.

Unable to list all the existing resources

The number of items returned in a single response from resources, like virtual machines or volumes, is limited to 1000.

Causes:

The API or user interface only lists 1000 resources. This is an intentional limit as a larger result sets require greater cost to derive and manage.

Solution:

If it is desired to see a larger result sets, increase the maximum number of instances returned in a single response by setting the osapi_max_limit property in
the /etc/nova/nova.conf file in the Compute Nodes of the related region. This will impact several interfaces, including the Nova list interface, the Horizon administrative user interface, and the IBM Cloud Orchestrator user interface. To manage all instances in a region, the recommended setting would be the maximum number of instances for the region and a growth buffer. For example, if the region can contain 2000 instances, and a 10% growth buffer is desired, a limit of 2200 should be used.

Cannot configure one nova compute service to connect too many vCenter clusters ( >4 )

This topic describes why you cannot configure a nova compute service to connect too many vCenter clusters.

According to the length of metadata column is 255 of instances table, you cannot configure the nova compute service to connect too many vCenter clusters, otherwise the VMware discovery tool may fail to discover the instances from vCenter with following error:


If you find this error in the /var/log/nova/discovery.log file, follow the procedures described in "Configuring vmware-discovery for multiple vCenters or cluster/resource pool" on page 126 and "Connecting to multiple clusters" on page 118.

Failed to start a virtual system instance on a KVM Region

A virtual system instance failed to start on a KVM Region.

Problem

When starting an instance running on a KVM Region from the Self-service user interface (PATTERNS > Instances > Virtual System Instances), the instance hangs in Launching status even if the image is in ACTIVE status in the nova list.

Causes

KVM Compute Node has the hardware clock configured as LOCAL and not as UTC.

Solution

Perform the following steps on all the KVM Compute Nodes in the region:

1. Make sure that hardware clock is treated as UTC by running the following command:
   ```bash
cat /etc/adjtime
   
   619.737272 1411131873 0.000000
   1411131873
   UTC
   ```

2. Make sure that the current system clock is synchronized with the other Compute Nodes, Central Servers, and Region Servers.

3. Save the current system clock to hardware by running the following command:
   ```bash
   hwclock --systohc
   ```
Troubleshooting for a VMware Region

There are some known problems and limitations that might occur in a VMware Region.

**Cannot find attached volume after volume-attach:**

The `nova volume-attach` can be used to attach a volume to an instance. Sometimes, the `volume-attach` command runs successfully, but when you run the `fdisk -l`, you cannot find the attached volume. After you restart the virtual machine, the volume is found. It is an known issue for a VMware hosted system. There are some workarounds you can use to discover the attached volume without rebooting the guest operating system, such as logging into the guest operating system and running the following command:

```
echo "- - -" > /sys/class/scsi_host/host#/scan
```

**A VMware instance is shut down every time it is restarted**

OpenStack has a feature where power states are synced between the OpenStack database and the managed hypervisors. If OpenStack records a virtual machine as shut down, it ensures that the virtual machine is shut down on the hypervisor as well. So, if a cluster inside an vCenter is managed by more than one OpenStack VMware Region, when a virtual machine is started from a VMware Region, the other region tries to stop it. To avoid this, make sure that each cluster of vCenter is managed by only one OpenStack VMware Region. When you configure a VMware Region, consider the following issues:

- You cannot share a vCenter cluster across IBM Cloud Orchestrator installations.
  - When you do, and then stop a virtual machine, whenever you start it, the other OpenStack ensures that it is stopped within 60 seconds.
- You cannot host the managed-from environments on the managed-to vCenter.
- Avoid out of band operations.

Troubleshooting Workload Deployer

You can troubleshoot problems in the Workload Deployer component

**Failure to deploy a virtual system or application when a linked clone is enabled**

There might be a failure to deploy a virtual system or a virtual application when a linked clone is enabled.

This happens when you use a flavor with key `VMWareUsedLinkedClones=true`. The deployment fails if the disk defined in flavor is not the same as the disk size in the VMware template. This is a VMware limitation that does not allow you to resize the linked disk.

**Increasing the default timeout if hypervisor fails**

If a hypervisor goes into a failure state, it might mean that you need to increase the Workload Deployer timeout value.

You can increase the default timeout value that determines how long Workload Deployer waits for the background hypervisor data to be obtained. The value can
be configured inside the following file: `/opt/ibm/rainmaker/purescale.app/private/expanded/ibm/rainmaker.cloud-4.0.0.1/config/zero.config`

The entry that requires a value change is: `/config/cloudburst/vmsupport/timeoutsmsOPENSTACK`.

The value is specified in milliseconds. The default value is 3600000 (= 1 hour).

---

**Troubleshooting a high-availability installation**

Troubleshoot problems that might occur when running a high-availability installation of IBM Cloud Orchestrator.

**Troubleshooting System Automation for Multiplatforms controlled services**

To provide high availability, the services on Central Server 2 and the Region Servers are clustered on two virtual machines and are controlled by System Automation for Multiplatforms. A service might be in an error state. You can check the state of these clusters in the following ways:

- Use the System Automation Application Manager web user interface: `<SAAM_server_hostname>:16311/ibm/console`.
- Log on to the command line of a cluster virtual machine and run the `lssam` command. The following sample output is for the Central Server 2 cluster:

```bash
Online IBM.ResourceGroup:central-services-2-rg Nominal=Online
  |- Online IBM.Application:bpm
    |- Online IBM.Application:bpm:cil021029163
    `- Online IBM.Application:bpm:cil021029164
  |- Online IBM.Application:haproxy
    `- Online IBM.Application:haproxy:cil021029163
      `- Offline IBM.Application:haproxy:cil021029164
  `- Online IBM.Application:ihs
    `- Online IBM.Application:ihs:cil021029163
      `- Offline IBM.Application:ihs:cil021029164
  `- Online IBM.Application:keystone
    `- Online IBM.Application:keystone:cil021029163
      `- Offline IBM.Application:keystone:cil021029164
  `- Online IBM.Application:sar_agent
    `- Online IBM.Application:sar_agent:cil021029163
      `- Offline IBM.Application:sar_agent:cil021029164
  `- Online IBM.Application:ui-rg Nominal=Offline
    |- Online IBM.Application:horizon
      `- Online IBM.Application:horizon:cil021029163
        `- Online IBM.Application:horizon:cil021029164
      `- Online IBM.Application:pcg
        `- Online IBM.Application:pcg:cil021029163
          `- Online IBM.Application:pcg:cil021029164
    `- Online IBM.Application:scui
      |- Online IBM.Application:scui:cil021029163
        `- Online IBM.Application:scui:cil021029164
      `- Online IBM.Application:scui:cil021029164
    `- Online IBM.ServiceIP:cs2-ip
      `- Online IBM.ServiceIP:cs2-ip:cil021029163
        `- Offline IBM.ServiceIP:cs2-ip:cil021029164
Online IBM.Equivalency:cs2-network-equ
  `- Online IBM.NetworkInterface:eth0:cil021029163
    `- Offline IBM.NetworkInterface:eth0:cil021029164
```

If a resource is not in the correct state (either offline or online), check the actual state of the service:
1. Log on, as a root user, to the virtual machine where the service runs. The host name of the VM is listed in the GUI or in the lssam output as property of the resource.

2. Run the following command:
   
   service <service> status

   **Tip:** The service registered in init.d might differ from the name of the application resource in System Automation for Multiplatforms. The output should match the state in the lssam command output.

3. If a service is in an error state or in unknown state, try to manually recover the service by running the following command on the virtual machine where the service should run:

   **Important:** If the service is an active/standby service, run this command on only one of the virtual machines of the cluster.

   service <service> stop; service <service> start

   For some services, you might encounter the following error situations:
   
   • `<name of service>` dead but subsys locked: Manually start the service again. This recovers the service and correctly sets the status to started.
   
   • `<name of service>` dead but pid file exists: Manually stop the service again. This deletes the pid file and correctly sets the status of the service to stopped. If this does not help, delete the pid file manually.

4. If the service does not start, check the log files of the service and correct any problems found there.

5. You might need to suspend the automation to resolve an error condition. In a normal state, the automation automatically restarts a service even if the service is stopped manually. To suspend the automation, log on to one of the cluster virtual machines and run the following command:

   samctrl -M t

   After the error condition is resolved, start the service and verify the correct status. If the status is correct, run the following command to resume the automation:

   samctrl -M f

6. If the resource is still in error, run the following command to reset the resource in System Automation for Multiplatforms:

   resetsrc -s 'Name == "<service>"' IBM.Application

### Managing the placement of services

During a system failure, certain services are moved in the cluster. To identify where a service is running, use the lssam command. If a service is configured as active/standby, the command output shows that the server where the service is running is online, and the standby server is offline. For maintenance or certain recoveries, you might need to manually move a service in the cluster. You can move a service by running the following command on one of the servers in the cluster:

   rgreq -n <fqdn_of_the_node_to_be_moved_from> -o move <name_of_resource_group_for_service>
Troubleshooting agentless adapter controlled services

If an agentless adapter resource is in error, complete the following steps:

1. Log on to the System Automation Application Manager web user interface:
   
   `<SAAM_server_hostname>:16311/ibm/console/`

   Specify the user and password that you defined during the installation. The default user is `eezadmin`.

2. Click the **Explore Automation Domains** tab. Expand the **SCOsaam** end-to-end automation policy.

3. Find the automation domain for the resource that is in error, and select this domain. The right pane shows the resources of this automation domain.

4. Right-click on the resource that is in error, and click **Reset**. After some time, the screen refreshes: if the resource recovered from the error, the resource is working correctly.

5. To understand the cause of the problem, check the log files. Right-click on the automation domain, and click **View Log**.

6. If the resource cannot recover from the error, log on to the server where the service is running. Check the status of the service in the context of the `saamuser` user by running the following command:

   ```bash
   su - saamuser -c "/opt/IBM/tsamp/eez/scripts/servicectrl.sh <service> status; echo $?"
   ```

   This command returns a System Automation Application Manager type return code. Check the error cause and correct it manually.

Using `/etc/hosts`

If the `/etc/hosts` file is used for System Automation Application Manager to resolve the host name of the IBM Cloud Orchestrator servers, you must restart System Automation Application Manager after each change in the `/etc/hosts` file to allow System Automation Application Manager to use the changes.

Suspending automation for a resource or a resource group for manual interaction

If you need to manually interact with an IBM Cloud Orchestrator management stack resource or resource group, you must stop the System Automation Application Manager automation to ensure that System Automation Application Manager does not try to bring the resource to the desired state. For example, if you are upgrading the product or installing a fix, you might need to start and stop a resource. To suspend the System Automation Application Manager automation, complete the following steps:

1. Log on to the System Automation Application Manager web user interface:
   
   `<SAAM_server_hostname>:16311/ibm/console/`

   Specify the user and password that you defined during the installation. The default user is `eezadmin`.

2. Click the **Explore Automation Domains** tab. Expand the **SCOsaam** end-to-end automation policy.

3. In the **Resources** section in the right pane, expand the list of resources and resource groups.

4. Right-click the resource or resource group for which you want to temporarily suspend the System Automation Application Manager automation, and click
Suspend Automation.... In the displayed window, click Submit. The resource or the resource group and all the resources in the group are excluded from the System Automation Application Manager automation.

5. After you finish manually interacting with the resource or the resource group, resume the automation by clicking Resume Automation. System Automation Application Manager brings the resource or the resource group to the desired state.

Troubleshooting Business Process Manager

There are some known issues and limitations that might occur when you are working with Business Process Manager.

Log tracing

In Business Process Manager, tracing is switched off by default. If you must troubleshoot or debug any issues, switch tracing on.

The SCOrchestrator toolkit saves in the log file all parameters passed by other toolkits used by runbooks. If these parameters contain security-sensitive information, they are visible in the `<path to BPM profile>/log/SingleClusterMember1/SystemOut.log` file.

To disable logging from the SCOrchestrator toolkit, perform the following steps:

1. Create a script file named disableLogging.py with the following content:
   ```python
   AdminTask.setTraceSpecification('[-persist true -traceSpecification *=info:WLE.wle_javascript=audit ]');
   AdminConfig.save();
   ```

2. Run the following command on Central Server 4 from the directory where file has been saved:
   ```bash
   <path to BPM profile>/bin/wsadmin.sh -host `uname -n` -username admin
   -password <passw0rd> -f disableLogging.py
   ```

To re-enable logging in the SCOrchestrator toolkit, perform the following steps:

1. Create a script file named enableLogging.py with the following content:
   ```python
   AdminTask.setTraceSpecification('[-persist true -traceSpecification *=info ]');
   AdminConfig.save();
   ```

2. Run the following command on Central Server 4 from the directory where file has been saved:
   ```bash
   <path to BPM profile>/bin/wsadmin.sh -host `uname -n` -username admin
   -password <passw0rd> -f enableLogging.py
   ```

The size of the Performance Data Warehouse Database increases continuously (> 100 GB) when IBM Cloud Orchestrator is running and finally results in out of space errors.

The growth of the Performance Data Warehouse Database is caused by error messages written to the table LSW_DATA_TRANSFER_ERRORS:

"undefined tracking group with external ID 5cd1ff9-d1ab-4a88-8810-b0e7dce571e" ...and so forth

Solution:

2. Open Process Apps or Toolkits, for example:
   ```
   SCOrchestrator_Support_vSys_Toolkit (SCOVSYS)
   ```
3. Click File > Update Tracking Definitions.

The installation of Central Server 2 fails and in the log file there are chef errors when installing Business Process Manager

Solution:

To resume the installation, remove Business Process Manager from Central Server 2. For information about removing Business Process Manager, see the Business Process Manager section of the IBM Knowledge Center. After Business Process Manager is removed, you can resume the installation with following command:

ds job-execute -r TRUE
Notices

This information was developed for products and services offered in the U.S.A.

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 non-IBM 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
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: 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 states 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 herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish 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 Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758
U.S.A.

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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM 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 non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All 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 the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.
Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered 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.
Glossary

This glossary includes terms and definitions for IBM Cloud Orchestrator.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to [www.ibm.com/software/globalization/terminology](http://www.ibm.com/software/globalization/terminology) (opens in new window).

**A**

**account code**
A code that uniquely identifies an individual, billing, or reporting entity within chargeback and resource accounting.

**account code conversion table**
An ASCII text file that contains the definitions that are required to convert the identifier values defined by the account code input field to the user-defined output account codes.

**account report**
A report that is used to show account level information for usage and charge.

**audit data**
A data record that contains information about specific types of user activity, security events, and configuration changes in the product and in the cloud.

**availability zone**
A logical group of OpenStack Compute hosts. It provides a form of physical isolation and redundancy from other availability zones, such as by using separate power supply or network equipment.
Bill program
A program that performs cost extensions within SmartCloud Cost Management and summarizes cost and resource utilization by account code. The Bill program uses the rate code table that is assigned to the client to determine the amount to be charged for each resource consumed.

building block
The model of an image that is created by combining models of a base operating system and software bundles. Each building block contains a semantic and functional model that describes the contents of the components, for example, the installed products, supported operating systems, prerequisites, and requirements.

business object
A software entity that represents a business entity, such as an invoice. A business object includes persistent and nonpersistent attributes, actions that can be performed on the business object, and rules that the business object is governed by.

business process
A defined set of business activities that represent the required steps to achieve a business objective. A business process includes the flow and use of information and resources.

chargeback identifier
A label, which is often tied to an algorithm or set of rules, that is not guaranteed to be unique, but is used to identify and distinguish a specific chargeback item or chargeback entity from others.

custom node
A virtual image part that provides an unconfigured node for a pattern that has a deployment manager or a control node as its base.

conversion mapping
An entry in a mapping table which allows you to map identifiers to accounts or other identifiers.

exception file
A file that contains a list of records with identifier names that do not have a matching Parameter IdentifierName attribute value.

exception processing
A process in which the system writes all records that do match an entry in the account code conversion table to an exception file.

human service
An activity in the business process definition that creates an interactive task that the process participants can perform in a web-based user interface.

hypervisor
Software or a physical device that enables multiple instances of operating systems to run simultaneously on the same hardware.

kernel
The part of an operating system that contains programs for such tasks as input/output, management and control of hardware, and the scheduling of user tasks.
parameter (parm)
A value or reference passed to a function, command, or program that serves as input or controls actions. The value is supplied by a user or by another program or process.

parm See parameter

performance counter
A utility that provides a way for software to monitor and measure processor performance.

primary key
In a relational database, a key that uniquely identifies one row of a database table.

process application
A container in the Process Center repository for process models and supporting implementations. A process application typically includes business process definitions (BPDs), the services to handle implementation of activities and integration with other systems, and any other items that are required to run the processes. Each process application can include one or more tracks.

proration
A process that distributes the overall or individual resources of an account and the cost of those resources across multiple accounts at a specified percentage.

proration table
An ASCII text file that defines the identifier values and rate codes that are used in the proration process.

rate code
The identifier of a rate that is used to link a resource unit or volume metric with its charging characteristics.

rate group
A group of rate codes that is used to create rate subtotals in reports, graphs, and spreadsheets.

registry
A repository that contains access and configuration information for users, systems, and software.

S

service operation
A custom operation that can be run in the context of the data center. These operations are typically administrative operations and are used to automate the configuration. Service operations can also be used to enhance the catalog of available services with extra functionality.

shared service
A predefined virtual application pattern that is deployed and shared by multiple application deployments in the cloud, including virtual applications, virtual systems, and virtual appliances.

software bundle
A collection of software installation files, configuration files, and metadata that can be deployed on a virtual machine instance.

T

toolkit
A container where artifacts can be stored for reuse by process applications or other toolkits.

V

virtual application
A complete set of platform resources that fulfill a business need, including web applications, databases, user registries, messaging services, and transaction processes. A virtual application is defined by a virtual application pattern. See also virtual application pattern

virtual application pattern
A pattern that defines the resources that are required to support virtual applications, including web applications, databases, user registries, and more. These patterns are the deployment unit for a virtual application. See also virtual application

virtual machine (VM)
An instance of a data-processing system that appears to be at the exclusive disposal of a single user, but whose
functions are accomplished by sharing the resources of a physical data-processing system.

**virtual system**
A collection of virtual machines.

**virtual system instance**
The virtual environment that runs on a hypervisor in the cloud.

**virtual system pattern**
One or more virtual images, which can include script packages, that implement a deployment topology. A virtual system pattern is a shared topology definition used for repeatable deployment.

**VM** See [virtual machine](#)
Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name, or other personally identifiable information for purposes of session management, enhanced user usability, single sign-on configuration. These cookies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

Accessibility features for IBM Cloud Orchestrator

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features of IBM Cloud Orchestrator are described in this topic.

Accessibility features

The following list includes the major accessibility features in IBM Cloud Orchestrator:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

**Note:** The default configuration of JAWS screen reader does not read tooltips. JAWS users must enable their current mode to read tooltips by selecting **Utilities > Settings Center > Speech Verbosity > Verbosity Level > Configure Verbosity Levels**.

User documentation is provided in HTML and PDF format. Descriptive text is provided for all documentation images.

The knowledge center, and its related publications, are accessibility-enabled.

Related accessibility information

You can view the publications for IBM Cloud Orchestrator in Adobe Portable Document Format (PDF) using the Adobe Reader. PDF versions of the documentation are available in the information center.

IBM and accessibility

See the [IBM Human Ability and Accessibility Center](#) for more information about the commitment that IBM has to accessibility.