IBM Content Navigator

Deploying IBM Content Navigator in a container

Contents

Introduction

This document provides steps for configuring and deploying an IBM Content Navigator (ICN) container.

Some of the steps that are involved contain terminology, parameters or values that are not related to IBM Content Manager (CM8) and Content Manager OnDemand (CMOD). However, the steps are essential for the deployment and need to be included. The steps that can be ignored are identified.

An LDAP server is required for the container deployment. LDAP is not required for CM8 and CMOD servers. This requirement is planned to be made optional or removed for these servers in a future version of the ICN container product.

In this document, content services container is used to indicate the ICN container. The references are made to any containerized software that provides content services.

The reader is assumed to be familiar with terms and concepts related to container software such as Docker, Kubernetes and Red Hat OpenShift. For more information, see:

- Kubernetes Concepts
- Getting started with Red Hat OpenShift on IBM Cloud

Preparing the environment for container deployments

When you prepare your environment, record the settings for the databases, LDAP, storage, logging and monitoring choices, and so on. You need these values available to enter into the custom resource YAML file for deployment and configuration.

About this task

For lists of the parameters that you need to collect, see the following section: “Configuration reference” on page 47.

For a new IBM Content Navigator environment, a database installation and storage are required. Prepare these prerequisites before you begin the configuration and deployment of your containers.

Understanding custom resources

The IBM Content Navigator container environment is deployed using an operator.

The following topics provide important information on the Kubernetes concepts that are employed by the operator to install and manage the IBM Content Navigator container.

Using an operator

The operator captures the expert knowledge of administrators on how the system ought to behave, how to deploy it, and how to react if problems occur.

The operator is built from the Red Hat and Kubernetes Operator Framework, which is an open source toolkit that is designed to automate features such as updates, backups, and scaling. The operator handles upgrades and reacts to failures automatically.

Operators help you to take care of repeatable tasks by using the Kubernetes APIs and `kubectl` tools. Operators "watch" over a Kubernetes environment and use its actual state versus the defined state to decide what actions to take.

The operator uses a custom resource definition (CRD), which describes what the operator is meant to watch. The `crd.yaml` file contains the description of the custom resources for the container images, and the `role.yaml` and `role_binding.yaml` files define the access to the resources. The `service_account.yaml` file creates a service account with a role that has the permissions to manage the resources. The CRD specifies a configuration, but the cluster also needs controllers to monitor its state and reconcile the resource to match with the configuration. When the CRD is deployed it can then be used to control the automation containers by using Kubernetes primitives such as **`Services`**, **`ReplicaSets`**, **`DaemonSets`**, and **`Secrets`**.

To control access, registry secrets and security context constraints (SCC) are needed to set the range of allowed IDs. A shared persistence volume (PV) is needed to store files such as JDBC drivers and other files that are used by the roles.

To install the content services components, you use capability-specific custom resource (CR) templates that are on GitHub. You then create and customize your CR file, which you then apply to the operator to watch.

Custom resource template structure

The custom resource template files are divided into sections, which define the attributes of each component. The custom resource **`metadata`** along with the shared, LDAP, and data source configuration sections in the specification (**`spec`**) are always present.

Two types of template are provided for enterprise deployment:

Simplified custom resource templates
A streamlined and condensed custom resource YAML file is provided with fewer parameters to specify. When you use the simplified YAML, ibm_fncm_cr_enterprise.yaml, the operator automatically supplies many default values for a deployment.

Fully customizable custom resource templates
If you want more control over the details of their deployment can continue to use the full custom resource YAML file, ibm_fncm_cr_enterprise_FC_content.yaml. This template includes the component-level configuration sections with default values for all of the parameters. These values are the default values that the operator uses with the simplified templates.

Note: All simplified CR templates are well-formed CR YAML files for the capabilities they define. Parameters that need custom values have a value of <Required>, which you must update with a real value before you apply the CR.

Your custom resource file that you apply to the operator must include the necessary sections. The sections define specific parts of your intended deployment.

The following snippets show how the sections are used to compile a custom resource file, with sample content.

1. The **metadata** section applies to all of the included capabilities.

```
apiVersion: fncm.ibm.com/v1
kind: FNCMCluster
metadata:
  name: fncmdeploy
  labels:
    app.kubernetes.io/instance: ibm-fncm
    app.kubernetes.io/managed-by: ibm-fncm
    app.kubernetes.io/name: ibm-fncm
    release: 5.5.5
```

2. The **spec** section defines the target platform, the deployment type, the license, security, monitoring, logging, and storage information.

```
spec:
  ################################################################################
  ## This section contains the shared configuration for all FNCM components #
  ################################################################################
  appVersion: 20.0.2
  shared_configuration:

    ## This is the deployment context is FNCM. No update it required.
    sc_deployment_context: FNCM
```

3. If you plan to use an identity provider for user management, the open_id_connect_providers section defines connection information.

```
    # open_id_connect_providers:
    ## Set a provider name that will be used in your redirect URL.
    #- provider_name: ""
      ## Set a display name for the sign in button in Navigator.
    #  display_name: "Single Sign on"
    #  # Enter your OIDC secret names for Content Platform Engine, Navigator, External Share
and GraphQL.
    #  # Not all secrets are required depending on your deployment. Specify secret only for
components that you are deploying.
```

4. The ldap_configuration and datasource_configuration sections define parameters for all of the included capabilities.

```
## The beginning section of LDAP configuration for FNCM
  ldap_configuration:
    ## The possible values are: "IBM Security Directory Server" or "Microsoft Active
Directory"
    lc_selected_ldap_type: "<Required>"

    ## The name of the LDAP server to connect
    lc_ldap_server: "<Required>"
```

```
    ## The database configuration for ICN (Navigator) - aka BAN (Business Automation
Navigator)
    dc_icn_datasource:
      ## Provide the database type from your infrastructure.  The possible values are "db2"
or "db2HADR" or "oracle" or "sqlserver" or "postgresql".
```

A simplified custom resource template does not include component configuration sections. All settings are provided as defaults by the operator.

The full custom resource template includes configuration sections for each component with default values provided. You can specify your own values as needed in this template.

If you want to customize individual components, then you can add the component configuration parameter (**xxx_configuration**) along with the list of parameters that you want to customize. Templates are provided with the full list of configuration parameters for each component. Each component has a clear section header that highlights where the set of parameters starts.

Tip: If you plan to add a component section to your simplified template, copy the entire header from the full configuration templates into the compiled custom resource so that you can identify where the component configuration starts and finishes.

```
################################################################################
#######        Business Automation Navigator configuration       #######
################################################################################
navigator_configuration:
   ...
```

Preparing your cluster

Set up your cluster and install the necessary software before you prepare your application environment for installing the containers with the operator.

Before you begin

Install the following list of software before you install the content servics container (ICN Container). containers.

- Kubernetes 1.11+.
- Kubernetes CLI. For more information, see https://kubernetes.io/docs/tasks/tools/install-kubectl/.
- The OpenShift Container Platform CLI has commands for managing your applications, and lower-level tools to interact with each component of your system.

  Refer to the OpenShift 3.11 documentation.

  Refer to the OpenShift 4.2 or higher documentation.
- All the container images require persistent volumes (PV) and persistent volume claims (PVCs), so review the topics on preparing these PVs, PVCs, an LDAP, and creating a database for storing ICN configuration data, for your intended or target installation.

Procedure

1. Install the necessary software and make sure that your environment is compatible with Cloud Native Computing Foundation (CNCF) Certified Kubernetes.

   If you are not sure which Certified Kubernetes platform is right for you, see Picking the right solution.

   The Detailed system requirements page provides a cluster requirements guideline for content services containers.
2. Use the download doc to see the list of eAssembly images for Kubernetes.

3. Create a namespace in your OpenShift environment through using **oc** commands.
   a) Log in to your cluster.

```
$ oc login https://<CLUSTERIP>:8443 -u <ADMINISTRATOR>
```

   b) Create an OpenShift project (namespace) in which you want to install the operator.

```
$ oc new-project my-project
```

Preparing your cluster on IBM Cloud (ROKS)

Set up your cluster on IBM Cloud or Red Hat OpenShift Kubernetes Service (ROKS) and install the necessary software before you prepare your application environment for installing the containers with the operator.

About this task

The scripts and Kubernetes descriptors in the GitHub repository are needed to install the containers.

Before you deploy an automation container on IBM Cloud, you must configure your client environment, create an OpenShift cluster, prepare your container environment, and set up where to get the container images.

Make sure that you have the following list of software on your computer so you can use the command-line interfaces (CLIs) you need to interact with the cluster.

- IBM Cloud CLI
- OpenShift Container Platform CL
- Kubernetes CLI
- Docker CLI (Mac) or Docker CLI (Linux)

As an administrator of the cluster you must be able to interact with your environment.

1. Create an account on IBM Cloud.
2. Log in to IBM Cloud if you already have an account.

If you do not already have a cluster, then create one. From the IBM Cloud Overview page, in the OpenShift Cluster tile, click **Create Cluster**. Refer to the IBM Cloud documentation to create a Kubernetes cluster. The cluster that you create includes attached storage.

Preparing the IBM Content Navigator database

Before you configure and deploy your IBM® Content Navigator container, prepare the required database that the application uses. This task applies only when you are preparing to deploy containers as part of a new installation. Otherwise, use your existing database.

About this task

When you prepare your environment, record the settings so that these values are available to enter into the custom resource YAML file for deployment and configuration. For lists of the parameters that you need to collect, see “Configuration reference” on page 47.

Procedure

- Prepare the IBM Content Navigator database:

  Db2

  For details about configuring a Db2 database for your IBM Content Navigator deployment, see Creating a Db2 database for Navigator

Oracle
For details about configuring an Oracle database for your IBM Content Navigator deployment, see Creating an Oracle database for Navigator.

Microsoft SQL Server

You create both a database and a schema for the container deployment, for example:

```
create database <dbName>
** connect to database **
create schema <schemaName>
```

For details about configuring a Microsoft SQL Server database for your IBM Content Navigator deployment, see https://www.ibm.com/support/knowledgecenter/SSEUEX_3.0.8/com.ibm.installingeuc.doc/eucin012.htm.

PostgreSQL
Set up your PostgreSQL database and record the database settings. For more information, see Setting up a PostgreSQL database and Creating secrets to protect sensitive PostgreSQL SSL configuration data.

Creating secrets to protect sensitive configuration data

Before you deploy, create secrets manually to protect the configuration data you are going to enter.

Procedure

1. Prepare your security environment:

   You must also create a secret for the security details of the LDAP directory and data sources that you configured in preparation for use with IBM Business Automation Navigator. Collect the users and passwords to add to the secret. Using your values, run the following command:

   ```
   kubectl create secret generic ibm-ban-secret \
     --from-literal=navigatorDBUsername="user_name" \
     --from-literal=navigatorDBPassword="xxxxxxx" \
     --from-literal=keystorePassword="xxxxxxx" \
     --from-literal=ltpaPassword="xxxxxxx" \
     --from-literal=appLoginUsername="user_name" \
     --from-literal=appLoginPassword="xxxxxxx"
     --from-literal=jMailUsername="mailadmin"
    --from-literal=jMailPassword="{xor}GDoxNiosbg=="
    -n "{{ namespace }}"
   ```

   The secret you create, `ibm-ban-secret`, is the value for the parameter `ban_secret_name`.

   Note: The `jMailUsername` and `jMailPassword` values are for enabling the Sendmail capability in Navigator.

2. Configure the root ca secret and trusted certificate list.

   The custom YAML file also requires values for the `root_ca_secret`

   and `trusted_certificate_list` parameters. The TLS secret contains the root CA's key value pair. You have the following choices for the root CA:

   - You can generate a self-signed root CA
   - You can allow the operator (or ROOTCA ansible role) to generate the secret with a self-signed root CA (by not specifying one)
   - You can use a signed root CA. In this case, you create a secret that contains the root CA's key value pair in advance.

   The list of the trusted certificate secrets can be a TLS secret or an opaque secret. An opaque secret must contain a `tls.crt` file for the trusted certificate. The TLS secret has a `tls.key` file as the private key.

3. Create an `lc_bind_secret` for your LDAP configuration information:

Use the `lc_bind_secret` for your LDAP configuration details, including your external LDAP details for external share, if applicable.

The following command shows how to create the (ldap-bind-secret) secret with the needed user names and passwords. (This example includes credentials for the optional external LDAP method for external share, which might not apply in your environment.)

For Open Shift Cloud Platform:

```
oc create secret generic ldap-bind-secret \
    --from-literal=ldapUsername="cn=admin,dc=ibm,dc=edu" --from-
literal=ldapPassword="<yourLDAPPassword>" \
    --from-literal=externalLdapUsername="cn=admin,dc=ibm,dc=edu" --from-
literal=externalLdapPassword="<yourLDAPPassword>"
```

For certified Kubernetes:

```
kubectl create secret generic ldap-bind-secret \
    --from-literal=ldapUsername="cn=admin,dc=ibm,dc=edu" --from-
literal=ldapPassword="<yourLDAPPassword>" \
    --from-literal=externalLdapUsername="cn=admin,dc=ibm,dc=edu" --from-
literal=externalLdapPassword="<yourLDAPPassword>"
```

The secret that you create, ldap_bind_secret, is the value for `ldap_configuration.lc_bind_secret`.

Configuring storage for the content services environment

Storage is required that is external to the containers in the content services environment. You set up and configure storage to prepare for the container configuration and deployment.

About this task

Each component container requires shared persistent storage to manage configuration information, application working space, and logs. Additionally, you can decide to set up file store areas to store document content and index areas to build full text indexes.

Data storage in a Kubernetes cluster is handled by using volumes. For Kubernetes, a PersistentVolume (PV) is a piece of networked storage in a cluster that is provisioned by an administrator. A PersistentVolumeClaim (PVC) is a request for storage that is made by a user. For more information about persistent volumes, refer to the Kubernetes documentation for persistent volumes.

The storage classes and persistent volumes carry the details of the real storage, which is then made available for use by cluster users without having any knowledge of the underlying infrastructure. A cluster administrator defines and creates a persistent volume by providing the cloud infrastructure with the details of the implementation of the storage. That storage can be a number of different types including NFS or a cloud-provider-specific storage system.

You can configure storage for your container environment in the following ways:

Manually create storage volumes and volume claims

This method is required in versions 20.0.1 and earlier. Before deploying the content services containers, you must create a set of persistent volumes (PV) and persistent volume claims (PVC) to use with the deployment of each container. Persistent volumes are provisioned in a static way. A cluster administrator creates a number of persistent volumes. The persistent volumes carry the details of the real storage, which is then made available for use by cluster users.

Dynamically provision storage through the operator functions

The StorageClass resource object describes and classifies storage that can be requested, as well as provides a means for passing parameters for dynamically provisioned storage on demand. StorageClass objects can also serve as a management mechanism for controlling different levels of storage and access to the storage. Cluster administrators define and create the StorageClass objects that users can request without needing any intimate knowledge about the underlying storage volume sources.

The persistent volume framework provided by the cloud platform enables both these functionalities and allows administrators to provision a cluster with persistent storage.

For specifics related to the cloud platform you are deploying into, refer to the documentation given by the your cloud provider or cloud platform.

If dynamic provisioning is used, the deployment can optionally utilize three storage classes to meet the "slow", "medium", and "fast" storage for the Cloud Pak components. You provide the names of these storage classes using these parameters in the custom resource YAML file:

```
sc_slow_file_storage_classname:
sc_medium_file_storage_classname:
sc_fast_file_storage_classname:
```

To avoid complexity, you can use the same storage class for "slow", "medium", and "fast". However to avoid potential performance issues, that single storage class should target fast storage.

Combine manually created storage and dynamically provisioned storage

Storage can be provisioned using a mixture of static and dynamic. The operator uses the persistent volume claim names, of the default values, to determine if a claim already exists.

- If the claim does not exist, dynamic provisioning is used. The persistent volume claim names provided in the custom resource YAML are used when the claim is created.
- If the claim does exist, that claim is used when deploying.
- If static provisioning is used, the persistent volumes and persistent volume claims must be pre-created and the persistent volume claim name provided in the CR.

For the content services containers, the persistent volume claims fall into four general categories:

- Configuration information shared by all container instances, or replicas, for a particular component. This storage experiences minimal accesses or changes. For that reason, a low quality of service (QoS) type of disk is acceptable. The operator will provision a PVC from the sc_slow_file_storage_classname.
- Logs with a potentially high number of writes and dynamic disk space requirements, where system performance benefits from disk with a higher quality of service. The operator will provision a PVC from the sc_medium_file_storage_classname.
- Application working space for a particular component. This storage, under certain workloads, experiences high rates of access. System performance benefits from disk with a higher quality of service. The operator will provision a PVC from the sc_fast_file_storage_classname.
- Data where the disk subsystem must meet a set of requirements for I/Os per second, space, backup power supplies, and so on, and have a long life expectancy. These disk subsystems might also be configured to meet high availability and recoverability requirements as needed by the organization. The operator will provision a PVC from the sc_fast_file_storage_classname.

The storage volumes and storage classes that you create must specify the appropriate reclaim policy and access modes:

- `accessModes: - ReadWriteMany`
- `persistentVolumeReclaimPolicy: Retain`

When you prepare your environment, record the settings so that these values are available to enter into the custom resource YAML file for deployment and configuration. For lists of the parameters that you need to collect, see “Configuration reference” on page 47.

Creating volumes and folders for deployment

The IBM Content Navigator container requires some persistent volumes, persistent volume claims, and folders to be created before you can deploy. The deployment process uses these volumes and folders during the deployment.

About this task

You can choose to have the operator dynamically provision storage for you at deployment time. In that case, you do not need to create volumes and folders manually for your container environment. You can leave all storage values with the default value in the custom resource YAML.

Remember: Storage can be provisioned using a mixture of static and dynamic. The operator uses the persistent volume claim names, of the default values, to determine if a claim already exists.

- If the claim does not exist, dynamic provisioning is used. The persistent volume claim names provided in the custom resource YAML are used when the claim is created.
- If the claim does exist, that claim is used when deploying.
- If static provisioning is used, the persistent volumes and persistent volume claims must be pre-created and the persistent volume claim name provided in the CR.

Although the following information describes the volumes that are generally required, you can decide to designate more or fewer persistent volumes and volume claims.

You can use a YAML file to capture details like the name and the specifications of the persistent volume that you want to create, and use the Kubectl command line tool with the file to create the persistent volume object. You use a similar approach to create the persistent volume claims. See the following example for more details: Configure a persistent volume for storage.

Directory permissions and ownership

The permissions that are described in the following steps are examples that provide a secure environment. Your environment might have different permission requirements. Consider the following possibilities when you apply permissions to your folders:

- The NFS export *root_squash* option is strongly recommended for security. If you use the root_squash option, then the file directories to be used for the PVs group ownership must be set to the one specified by the *anongid* option given in the NFS export definition. The default anongid value is 65534.
- If the no_root_squash option is used, the PV group ownership must be set to the root group 0.
- Assign read, write, execute permissions to both the user and group owners, for example, chmod 770

The following settings are required when creating your NFS exports:

- The *rw,sync,no_wdelay* settings are required.
- The *no_subtree_check* setting is recommended for performance.

Tip: When you replace the value for the <NFS Server> in the samples, you might need to provide the private IP of the server, depending on your environment.

Remember: The storage volumes that you create must specify the appropriate reclaim policy and access modes:

- `accessModes: - ReadWriteMany`
- `persistentVolumeReclaimPolicy: Retain`

The persistent volume and persistent volume claim names that are provided in the following tables are examples.

Procedure

- Create the persistent volumes and persistent volume claims for the IBM Content Navigator container deployment:

Table 1. Volumes, volume claims, and folders for IBM Content Navigator

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as seen by container
IBM Content Navigator Liberty configuration	`/configDropins/overrides`	icn-cfgstore-pv icn-cfgstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides`
IBM Content Navigator and Liberty logs	`/logs`	icn-logstore-pv icn-logstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/logs`
Custom plug-ins for IBM Content Navigator	`/plugins`	icn-pluginstore-pv icn-pluginstore-pvc	`/opt/ibm/plugins`
IBM Content Navigator viewer logs for Daeja® ViewONE	`/icnvwlogstore`	icn-vw-logstore-pv icn-vw-logstore-pvc	`/opt/ibm/viewerconfig/logs`
IBM Content Navigator storage for the Daeja ViewONE cache	`/icnvwcachestore`	icn-vw-cachestore-pv icn-vw-cachestore-pvc	`/opt/ibm/viewerconfig/cache`
IBM Content Navigator storage for Aspera® **Note:** This volume is required even though Aspera might not be used in your environment.	`/icnasperastore`	icn-asperastore-pv icn-asperastore-pvc	/opt/ibm/aspera

For each of the folders, set the ownership as shown in the following example using the default NFS anongid value:

```
chgrp -R 0 /icncfgstore
```

For each of the folders, change the permissions settings as follows:

```
chmod -Rf g=u /icncfgstore
```

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the IBM Content Navigator configuration store volume.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: icn-cfgstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/configDropin/overrides
    server: <NFS_SERVER>
```

```
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-cfgstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-cfgstore-pv
  volumeName: icn-cfgstore-pv
status:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
```

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the IBM Content Navigator and Liberty logs.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: icn-logstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/logs
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-logstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-logstore-pv
  volumeName: icn-logstore-pv
status:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
```

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the IBM Content Navigator plugins.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: icn-pluginstore-pv
spec:
  accessModes:
```

```
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/plugins
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cn-pluginstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-pluginstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-pluginstore-pv
  volumeName: icn-pluginstore-pv
status:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
```

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the IBM Content Navigator viewer logs.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: icn-vw-logstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/viewerlog
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-vw-logstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-vw-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-vw-logstore-pv
  volumeName: icn-vw-logstore-pv
status:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
```

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the IBM Content Navigator viewer cache store.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: icn-vw-cachestore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/viewercache
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-vw-cachestore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-vw-cachestore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-vw-cachestore-pv
  volumeName: icn-vw-cachestore-pv
status:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
```

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for Aspera.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: icn-asperastore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/aspera
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-asperastore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-asperastore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-asperastore-pv
  volumeName: icn-asperastore-pv
status:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
```

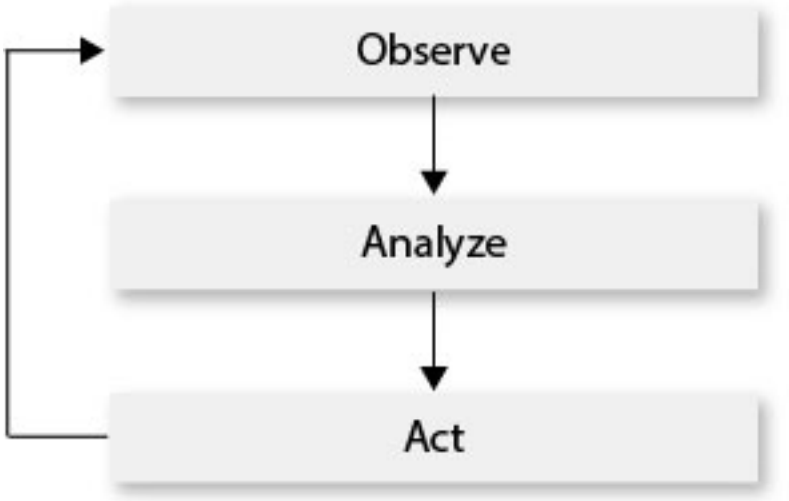
Deploying the operator

Operators make it simpler to install and update without having to worry about the underlying cloud provider. The operator captures the expert knowledge of administrators on how the system ought to behave, how to deploy it, and how to react if problems occur.

About this task

The operator is built from the Red Hat and Kubernetes Operator Framework, which is an open source toolkit that is designed to automate features such as updates, backups, and scaling. The operator handles upgrades and reacts to failures automatically.

Operators help you to take care of repeatable tasks by using the Kubernetes APIs and `kubectl` tools. Operators "watch" over a Kubernetes environment and use its actual state versus the defined state to decide what actions to take. The following diagram shows the control loop that occurs as a result of constantly watching the state of the Kubernetes resources.



The following concepts are key to managing operators:

Namespace deployment

At the beginning of a project, an administrator might encourage developers to use a short-lived namespace that is in their own name. These namespaces do not need to be metered. In the long term and for tracking purposes, it makes sense to divide up workloads into dedicated namespaces for your application lifecycle stages, such as development, preproduction, and production.

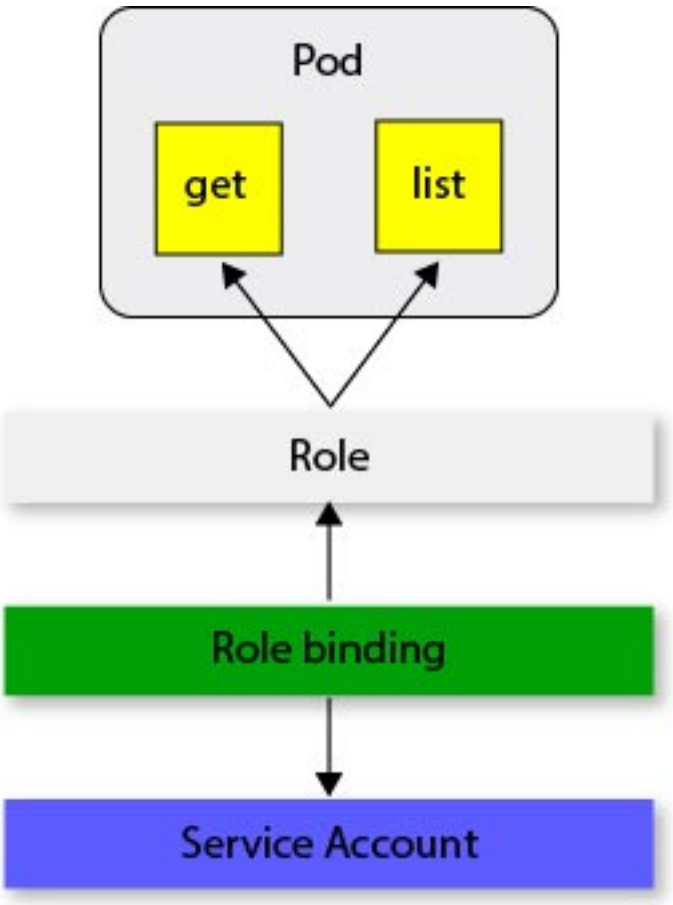
You can meter your deployments by the namespace and the platform label or **`swidtag`**. Metering uses a system of metadata annotation and aggregation of VPCs, and is critical to help you understand your deployments against entitlements.

Roles, role binding, and service accounts

The default for role-based access control (RBAC) is to deny all access unless you grant access.

A **`role`** scopes a set of operations that can be carried out on a group of resources. A **`service account`** provides an identity for processes that run in a pod, so it is necessary to grant a role to a service account for intra-cluster processes. A **`role binding`** associates a service account to a role, which in effect determines the operations that the account can run inside a namespace. An administrator can configure the role and role binding resources for each application before any operator is deployed. Each application must specify a `serviceAccountName` in its pod spec, and the service account must be created.

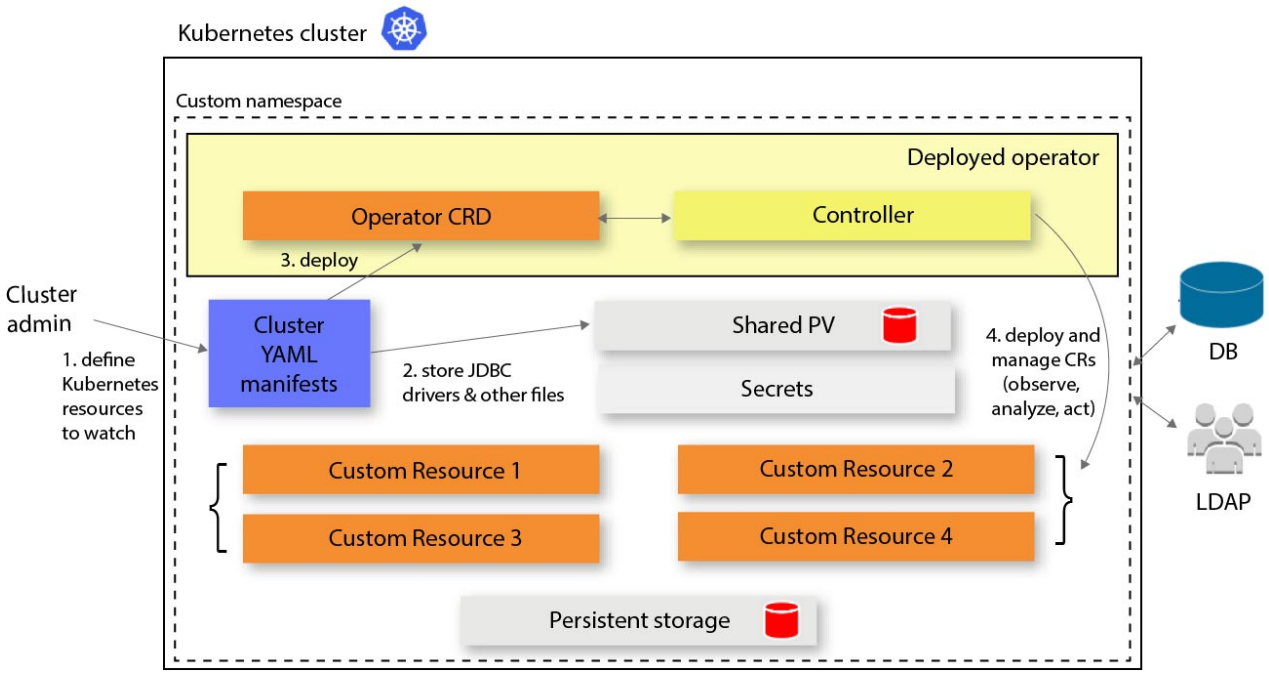
The following diagram shows the relationship between the objects that are used to grant users access to Kubernetes API resources.



The first step to use the operator is to create the new custom resource definition (CRD) that the operator is meant to watch. The `crd.yaml` file contains the description of the custom resources for the container images, and the `role.yaml` and `role_binding.yaml` files define the access to the resources. The `service_account.yaml` file creates a service account with a role that has the permissions to manage the resources. The CRD specifies a configuration, but the cluster also needs controllers to monitor its state and reconcile the resource to match with the configuration. When the CRD is deployed it can then be used to control the automation containers by using Kubernetes primitives such as **`Services`**, **`ReplicaSets`**, **`DaemonSets`**, and **`Secrets`**.

To control access, you must create registry secrets and security context constraints (SCC) to set the range of allowed IDs. A shared persistence volume (PV) is needed to store files such as JDBC drivers and other files that are used by the roles.

The following diagram shows the steps that are involved and highlights the key components in controlling a deployed operator.



Set up your local repository

Create a local copy of the Git Hub repository.

About this task

The Git Hub repository branch that you use changes depending on the version that you want to deploy. Specify the right repository version that contains the target level of ICN container that you wish to deploy. For example, repository version 5.5.5 maps to ICN Container v3.0.8

Procedure

1. Download or clone the repository on your local machine:

```
git clone -b 5.5.5 git@github.com:ibm-ecm/container-samples.git
```

2. Change to the operator folder in your local repository:

```
cd container-samples
```

Preparing storage for the operator

All instances of an operator need a place to store its log files whether it is on a private cloud or a public cloud.

Preparing the operator storage

All instances of an operator need a place to store its log files and find database drivers.

About this task

You must prepare the storage of the operator before you create an instance of the operator.

Procedure

1. Create a PV for the operator.

a. **Optional**: The following example YAML defines a PV, but PVs depend on your cluster configuration.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    type: local
  name: operator-shared-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/root/operator"
  persistentVolumeReclaimPolicy: Delete
```

b. **Optional**: Deploy the PV.

On OpenShift Cloud Platform:

```
oc create -f operator-shared-pv.yaml
```

On certified Kubernetes:

```
kubectl create -f operator-shared-pv.yaml
```

c. **Mandatory**: Provide group write permission to the persistent volume of the operator. According to the PV **hostPath.path** definition, run the following commands:

```
chmod -R g=u /root/operator
chmod g+rw /root/operator
```

Remove the `.OPERATOR_TYPE` file in case it exists from a previous deployment.

```
rm -f /<hostPath>/.OPERATOR_TYPE
```

Where *<hostPath>* is the value in your PV (`root/operator`).

2. Create a claim for the PV.

a) Create the claim:

(Dynamically) If you prefer to use dynamic provisioning for this claim, edit the provided file `descriptors/operator-shared-pvc.yaml` and replace the `<StorageClassName>` placeholder by a storage class of your choice.

(Manually) To create a claim bound to the previously created PV, create the file `<path>/operator-shared-pvc.yaml` anywhere on your disk, with the following content:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: operator-shared-pvc
  namespace: <MyProject>
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 1Gi
  volumeName: operator-shared-pv
```

Replace the `<MyProject>` placeholder with the name of your OpenShift project or namespace that you intend to use for your script-based or manual installation.

b) Deploy the PVC.

If you created your own `operator-shared-pvc.yaml`:

On OpenShift Cloud Platform:

```
oc create -f <path>/operator-shared-pvc.yaml
```

On certified Kubernetes:

```
kubectl create -f <path>/operator-shared-pvc.yaml
```

Otherwise, if you edited `descriptors/operator-shared-pvc.yaml`:

On OpenShift Cloud Platform:

```
oc create -f descriptors/operator-shared-pvc.yaml
```

On certified Kubernetes:

```
kubectl create -f descriptors/operator-shared-pvc.yaml
```

3. Add the JDBC drivers.

   Copy all of the JDBC drivers that are needed by the components you intend to install to the persistent volume. Depending on your storage configuration, you might not need these drivers.

   Note: File names for JDBC drivers cannot include more version information.

   - Db2
     - `db2jcc4.jar`
     - `db2jcc_license_cu.jar`
   - Oracle
     - `ojdbc8.jar`
   - Microsoft SQL
     - `mssql-jdbc<supported_version>.jar`
   - PostgreSQL
     - `postgresql-<supported_version>.jar`

   The following structure shows an example remote file system.

```
pv-root-dir
   └── jdbc
       ├── db2
       │   ├── db2jcc4.jar
       │   └── db2jcc_license_cu.jar
       ├── oracle
       │   └── ojdbc8.jar
       ├── mssql
       │   └── mssql-jdbc <supported_version>.jar
       ├── postgresql
       │   └── postgresql-<supported_version>.jar
```

Preparing the operator storage on IBM Cloud (ROKS)

All instances of an operator on IBM Cloud need a place to store its log files.

About this task

You can attach endurance storage with `gid` storage classes.

- `cp4a-file-retain-bronze-gid`
- `cp4a-file-retain-silver-gid`
- `cp4a-file-retain-gold-gid`

The YAML files to create these storage classes are provided in the descriptors folder.

To copy the JDBC drivers to the operator pod, you need to create a new storage class with one of these storage requirements.

Procedure

1. Create a storage class YAML file, and name it `operator-sc.yaml`.
2. Apply the new storage class.

   ```
   oc apply -f operator-sc.yaml
   ```

3. Create a claim for a PV dynamically by using descriptors/operator-shared-pvc.yaml.

   Replace the storage class with the name of the storage class from Step 1.
4. Deploy the PVC.

   ```
   oc create -f descriptors/operator-shared-pvc.yaml
   ```

5. Get the bound PV name and the PV location.

   ```
   oc get pvc | grep operator-shared-pvc
   oc describe PV PV_name
   ```

6. If your storage configuration needs JDBC drivers, create a `jdbc` parent folder on your remote file system and put your drivers into the following structure.

   ```
   pv-root-dir
      └── jdbc
         ├── db2
         │   ├── db2jcc4.jar
         │   └── db2jcc_license_cu.jar
         ├── oracle
         │   └── ojdbc8.jar
         ├── mssql
         │   └── mssql-jdbc <supported_version>.jar
         ├── postgresql
         │   └── postgresql-<supported_version>.jar
   ```

   Note: File names for JDBC drivers cannot include more version information.

   - Db2
     - `db2jcc4.jar`
     - `db2jcc_license_cu.jar`
   - Oracle
     - `ojdbc8.jar`
   - Microsoft SQL
     - `mssql-jdbc<supported_version>.jar`
   - PostgreSQL
     - `postgresql-<supported_version>.jar`
7. Copy these files to the operator pod.

   ```
   podname=$(oc get pod | grep ibm-cp4a-operator | awk '{print $1}')
   kubectl cp $PATH_TO_JDBC/jdbc $NAMESPACE/$podname:/opt/ansible/share -c ansible
   ```

Deploying a IBM Content Navigator container

The steps for deploying containers depend on the platform and configuration approach that you choose.

Before you begin

Before you start your container deployment process, review and complete the preparation tasks in “Preparing the environment for container deployments” on page 7.

About this task

After you prepare the environment, you use the configuration and deployment tools to deploy your content services containers in the environment of your choice. After the deployment, you perform additional setup steps to get your environment up and running.

Installing with an operator

You use the operator YAML manifest files in the GitHub repository to deploy an operator and use a custom resource YAML file to apply deployments of the content services containers.

Before you begin

You must prepare your environment and set up your cluster before you complete and deploy your custom resource YAML. For more information, see “Preparing the environment for container deployments” on page 7.

Getting access to container images

You must get access to the container images before you edit the custom resource file. The steps that you need to take depend on the platform that you plan to use.

Getting access to the container images

To get access to the container images, you must have an IBM entitlement registry key to pull the images from the IBM Docker registry or download the .tgz package file from Passport Advantage (PPA). If you download the images, you must push the images to a local docker registry.

About this task

The scripts and Kubernetes descriptors in the GitHub repository are needed to install the containers.

Procedure

1. In your local clone of the GitHub repository, go to the `container-samples` directory.
2. Get your entitlement key or download the images from PPA.
   - Option 1: Get your entitlement key for the IBM Cloud Entitled Registry.
     a. Log in to MyIBM Container Software Library with the IBMid and password that is associated with the entitled software.
     b. In the **Container software library** tile, verify your entitlement on the **View library** page, and then go to **Get entitlement key** to retrieve the key.
     c. Create a pull secret for the entitlement key, for example:

```
$ kubectl create secret docker-registry admin.registrykey --docker-server=cp.icr.io --
docker-username=cp --docker-password="<ENTITLEMENT_KEY_GENERATED>" --docker-
email=user@foo.com
```

Note: The `cp.icr.io` value for the `docker-server` parameter is the only registry domain name that contains the images. Use “cp” for the docker-username. The docker-email has to be a valid email address (associated to your IBM ID). Make sure you are copying the Entitlement Key in the docker-password field within double-quotes.

d. Take a note of the secret and the server values so that you can set them to the `pullSecrets` and `repository` parameters when you run the operator for your containers.

- Option 2: Download the packages from PPA and load the images.

  IBM Passport Advantage (PPA) provides archives (.tgz) for the software. To view the list of Passport Advantage eAssembly installation images, refer to the download document.

  a. Download one or more PPA packages to a server that is connected to your Docker registry.

  b. Log in to your cluster.

  c. Check that you can run a docker or podman command.

  ```
  docker ps
  ```

  ```
  podman ps
  ```

  d. Log in to the Docker registry with a token:

  ```
  docker login <registry url> -u <ADMINISTRATOR> -p <password>
  ```

  Note: You can connect to a node in the cluster to resolve the `docker-registry.default.svc` parameter.

  e. Run a `kubectl` command to make sure that you can use Kubernetes.

  ```
  kubectl cluster-info
  ```

  f. Run the `scripts/loadimages.sh` script to load the images into your Docker registry. Specify the two mandatory parameters in the command line.

  ```
  -p  PPA archive files location or archive file name
  -r  Target Docker registry and namespace
  -l  Optional: Target a local registry
  ```

  The following example shows the input values in the command line.

  ```
  cd scripts
  ./loadimages.sh -p <PPA-ARCHIVE>.tgz -r docker-registry.default.svc:5000/<project-name>
  ```

  Note: The `project-name` variable is the name of the project that you created when you set up your cluster. If you want to use an external Docker registry, take a note of the docker registry service name or the URL so that you can enter it during deployment. If you connect remotely to the cluster from a Linux host/VM, then you must have Docker and the OpenShift command line interface (CLI) installed on OCP. If you have access to the master node on the cluster, the OCP CLI and Docker are already installed.

  g. Check that the images are pushed correctly to the registry. Using the OpenShift CLI:

  ```
  oc get is
  ```

  h. In your target namespace, create a Docker registry secret if you want to use an external Docker registry or reuse a secret in the target project if you want to use an internal Docker registry.

  If you want to pull directly from the IBM entitled registry, reuse the secret that you created in Step 1 Option 1:

  ```
  imagePullSecrets:
     name: "admin.registrykey"
  ```

  Note: The secret_name must match the imagePullSecrets.name parameter in the operator deployment (.yaml) file, for example, admin.registrykey.

Create a secret to access an external Docker registry:

```
$ oc create secret docker-registry admin.registrykey --docker-server=<registry_url> --
docker-username=<your_account> --docker-password=<your_password> --docker-
email=fncmtest@ibm.com
```

For an internal Docker registry:

```
$ oc project <my-project>
$ oc get secret
```

3. In your target namespace, at deployment time, verify that the secret that you created for your image pull secret is still valid and has not expired. If needed, delete and recreate the secret as applicable in the previous steps.

Getting access to the container images on IBM Cloud (ROKS)

To get access to the container images on Managed Red Hat OpenShift Kubernetes Service (ROKS), you must have an IBM entitlement registry key to pull the images from the IBM docker registry or download the Cloud Pak package (.tgz file) from Passport Advantage (PPA). If you download the images, you must push the images to a local docker registry. The deployment script asks for the entitlement key or user credentials for the local registry.

Procedure

1. In your local clone of the GitHub repository, go to the `container-samples` directory.
2. Create a project for each release that you want to install by running the following command.

```
oc new-project <project_name> --description="<description>" --display-name="<display_name>"
```

3. Run a `kubectl` command to make sure that you have access to Kubernetes.

```
kubectl cluster-info
```

4. Make sure that your entitled container images are available and accessible in one of the IBM docker registries.
   - `Option 1`: Create a pull secret for the IBM Cloud Entitled Registry.
     a. Log in to MyIBM Container Software Library with the IBMid and password that is associated with the entitled software.
     b. In the **Container software library** tile, click **View library** and then click **Copy key** to copy the `entitlement_key` to the clipboard.
     c. Create a pull secret by running a `kubectl create secret` command.

```
kubectl create secret docker-registry <my_pull_secret> -n "<namespace>"
  --docker-server=cp.icr.io
  --docker-username=cp
  --docker-password="<entitlement_key>"
  --docker-email=user@foo.com
```

     Note: The `cp.icr.io` and `cp` values for the **`docker-server`** and **`docker-username`** parameters must be used. Take a note of the pull secret and the server values so that you can set them to the **`pullSecrets`** and **`repository`** parameters when you run the installation for your containers.

     d. Install the Container Registry plug-in.

```
ibmcloud plugin install container-registry -r 'IBM Cloud'
```

     e. Log in to your IBM Cloud account.

```
ibmcloud login -a https://cloud.ibm.com
```

f. Set the region as global.

```
ibmcloud cr region-set global
```

g. List the available images by using the following command.

```
ibmcloud cr image-list --include-ibm | grep -i cp4a
```

- `Option 2`: Download the packages from PPA and load the images

a. If you do not already have the images that you want to install, go to Passport Advantage and find the part numbers in the download document. Download the operator container and the IBM Content Navigator container.

b. Log in to your IBM Cloud account in the IBM Cloud CLI.

```
ibmcloud login --sso
```

Then, enter the one time code that is sent to your computer.

c. Log your local Docker daemon into the IBM Cloud Container Registry, create the project namespaces, list the new namespaces, and check that you can run docker.

```
ibmcloud cr login
ibmcloud cr namespace-add <project_name>
ibmcloud cr namespace-list
docker ps
```

Run a `kubectl` command to make sure that you can use the Kubernetes CLI.

```
kubectl cluster-info
```

d. Download the `loadimages.sh` script. Change the permissions so that you can run the script.

```
chmod +x loadimages.sh
```

e. Use the `loadimages.sh` script to push the images into the IBM Cloud Container Registry.

```
./loadimages.sh -p <PPA-ARCHIVE>.tgz -r <registry_domain_name>/<project_name>
```

Note: A registry domain name is associated with your cluster location. The name `us.icr.io` for example, is for the region **`us-south`**. The region and registry domain names are listed on the https://cloud.ibm.com/docs/services/Registry.

f. After you push the images to the registry, check whether they are pushed correctly by running the following command.

```
ibmcloud cr images --restrict <project_name>
```

g. Create a pull secret to be able to pull images from the IBM Cloud Container Registry.

```
kubectl --namespace <project_name> create secret docker-registry <my_pull_secret> \
   --docker-server=<registry_domain_name> --docker-username=iamapikey \
   --docker-password="<APIKEY>" --docker-email=<IBMID>
```

To generate an API KEY, go to **Security** > **Manage** > **Identity and Access** > **IBM Cloud API Keys** in the **IBM Cloud** menu and select **Generate an IBM Cloud API key**.

h. Take a note of the secret names so that you can set them to the **`pullSecrets`** parameter when you run the installation for your containers.

Deploying the operator

The YAML files and scripts that you need to deploy the operator are provided in the Git Hub repository.

About this task

The operator has a number of descriptors that must be applied.

- descriptors/fncm_v1_fncm_crd.yaml contains the description of the Custom Resource Definition.
- descriptors/operator.yaml defines the deployment of the operator code.
- descriptors/role.yaml defines the access of the operator.
- descriptors/role_binding.yaml defines the access of the operator.
- descriptors/service_account.yaml defines the identity for processes that run inside the pods of the operator.

Procedure

To deploy the operator:

1. In the `descriptors/operator.yaml` file, add license acceptance, modify the `image` parameter for containers (ansible and operator) to a valid image registry URL, and modify the `imagePullSecrets` name to the secret that you created when you prepared your cluster.

```
containers:
   - name: ansible
       # Replace this with the built image name
       image: "cp.icr.io/cp/cp4a/icp4a-operator:20.0.2"
  -
  -
  - name: operator
       # Replace this with the built image name
       image: "cp.icr.io/cp/cp4a/icp4a-operator:20.0.2"

       # MUST exist, used to accept fncm license, valid value only can be "accept"
         - name: fncm_license
           value:

   imagePullSecrets:
    - name: "admin.registrykey"
```

2. Deploy the operator on your cluster.

   The script deployOperator.sh can be used to deploy all of the descriptors and the operator pod.

```
$ ./scripts/deployOperator.sh -i <registry_url>/icp4a-operator:20.0.2 -p '<secret_name>' -n
<namespace>
```

   Note: For Open Shift Cloud Platform, if you do not specify the -i and -n options, the operator is deployed in the default namespace at this URL: `master_node:8500/default/icp4a-operator:20.0.2`. If you plan to use a non-admin user to install the operator, you must add the user to the `ibm-fncm-operator` role. For example:

```
$ oc adm policy add-cluster-role-to-user ibm-fncm-operator <user_name>
```

   If you want to deploy the operator YAML files without using the `deployOperator.sh` script, you can use the deploy command to deploy each file. You must manually update the parameter for license acceptance in the `operator.yaml` file before you deploy the files:

```
             - name: fncm_license
               value:
```

   Set the value to `accept`.

   To deploy each file on Open Shift Cloud Platform:

```
oc apply -f ./descriptors/fncm_v1_fncm_crd.yaml
oc apply -f ./descriptors/service_account.yaml
oc apply -f ./descriptors/role.yaml
oc apply -f ./descriptors/role_binding.yaml
oc apply -f ./descriptors/operator.yaml
```

To deploy each file on certified Kubernetes:

```
kubectl apply -f ./descriptors/fncm_v1_fncm_crd.yaml
kubectl apply -f ./descriptors/service_account.yaml
kubectl apply -f ./descriptors/role.yaml
kubectl apply -f ./descriptors/role_binding.yaml
kubectl apply -f ./descriptors/operator.yaml
```

3. Monitor the pod until it shows a STATUS of Running or Completed:

   For Open Shift Cloud Platform:

   ```
   $ oc get pods -w  | grep -v -E "(Running|Completed|STATUS)";do sleep 5;done
   $ oc logs -f <operator-pod> -c operator
   ```

   For certified Kubernetes:

   ```
   $ kubectl get pods -w  | grep -v -E "(Running|Completed|STATUS)";do sleep 5;done
   $ kubectl logs -f <operator-pod> -c operator
   ```

Deploying a custom resource

You create a custom resource file by editing one of the provided templates. You can then configure and customize the parameters of your selected capabilities in the custom resource, validate the file, and then apply it.

Creating the custom resource

All of the Kubernetes descriptors that are necessary to install manually can be found in GitHub. Components can be installed by creating a custom resource by hand and then running commands on the platform.

Before you begin

You must prepare your environment and cluster before you follow the manual installation instructions. For more information, see “Preparing the environment for container deployments” on page 7.

About this task

You can install IBM Content Navigator with default settings by using the simplified CR template with minimal customization. Comment out the sections for FileNet Content Manager and GraphQL.

Procedure

1. Make a copy of the custom resource YAML file that best reflects what you intend to deploy, either simplified or full custom, and name it for your deployment (`ibm_icn_my_cr_final.yaml`).
2. Copy and paste component sections into the YAML as needed, so that your YAML reflects all of the components that you want to deploy.

   Important: The `metadata.name` setting is used as a prefix for multiple objects. Because the maximum length of labels in Kubernetes is 63 characters, specify a short CR name. The total length of the CR name and an instance name must not exceed 24 characters, otherwise some component deployments fail. For example:

   ```
   metadata:
     name: icndeploy
   ```

What to do next

1. Configure the software that you want to install by using the instructions in "Checking and completing your custom resource" on page 31.
2. When your custom resource is complete, you can go to "Applying the custom resource" on page 34.

   Each time that you need to make an update or modification you must use this same file to apply the changes to your deployments. When you apply a new custom resource to an operator, you must make sure that all previously deployed resources are included. If you do not, the operator deletes them.

Checking and completing your custom resource

A custom resource YAML file is a configuration file that describes a container environment and includes the parameters to install IBM Content Navigator capabilities.

About this task

A single custom resource file is used to include all of the components that you want to deploy with an operator instance. Each time that you need to make an update or modification, you must use this same file to apply the changes to your deployments. When you apply a new custom resource to an operator, you must make sure that all previously deployed resources are included, otherwise the operator deletes them.

Checking the cluster configuration

You must check and edit the shared sections of the compiled custom resource file before you apply it to the operator.

About this task

Check the <Required> value for the **image_pull_secrets**, which are secrets in your target namespace to pull images from the specified repository. and **images** parameters in the **shared_configuration** section.

Check for **sc_image_repository**, which is a reference to your image registry.

Procedure

1. Locate the **shared_configuration** section in the custom resource (CR) file (`ibm_icn_my_cr_final.yaml`) you created in "Creating the custom resource" on page 30, then check and correct the deployment parameters.

   The custom resource templates can include the following parameters:

   License parameter

   - `ecm_configuration.cpe.license`: Change to "accept".

   Platform parameters

   - sc_deployment_platform, which can be OCP, ROKS, or blank.
   - sc_deployment_hostname_suffix, check the console route `canonicalName` for OCP.

   Note: If your target platform is ROKS, you need to get the IP address of the cluster sub-domain. On ROKS 3.11 clusters, multiple levels of sub-domains are not supported, so the **IBM-provided domain** does not work and the **Custom domain** option must be used. Run the following command to get the hostname :

   ```
   oc get route console -n openshift-console -o yaml|grep routerCanonicalHostname
   ```

   You must then ping the host to get the IP address. Enter the address in the sc_deployment_hostname_suffix parameter:

   ```
   sc_deployment_hostname_suffix:"{meta.namespace}.yourdomain.com"
   ```

Storage parameters

- sc_slow_file_storage_classname, is mandatory.
- sc_medium_file_storage_classname, is mandatory.
- sc_fast_file_storage_classname, is mandatory.

If you do not have three storage classes or you do not want to create them, you can use the same one for "slow", "medium", and "fast".

Content parameters

- sc_content_initialization, which can be: true or false.
- sc_content_verification, which can be: true or false.

2. Configure the root secret and trusted certificate list.

   The custom YAML file also requires values for the **root_ca_secret** and **trusted_certificate_list** parameters. The TLS secret contains the root CA's key value pair. You have the following choices for the root CA:

   - You can generate a self-signed root CA.
   - You can allow the operator to generate the secret with a self-signed root CA (by not specifying one).
   - You can use a signed root CA. In this case, you create a secret that contains the root CA's key value pair in advance.

   The list of the trusted certificate secrets can be a TLS secret or an opaque secret. An opaque secret must contain a `tls.crt` file for the trusted certificate. The TLS secret has a `tls.key` file as the private key.

3. Enter the parameter values for your LDAP instance in the **ldap_configuration** section.

   If you need to create a secret for the **lc_bind_secret** parameter to store the bind dn and bind password, then go ahead and create it.

   ```
   kubectl create secret generic my-ldap-tds-secret --from-literal=lc_ldap_bind_dn="cn=root" --
   from-literal=lc_ldap_bind_password="XXXXXXXX"
   ```

   Set the value in the custom resource file.

   Note: LDAP Anonymous authentication does not need a secret.

   If you want to use SSL-enabled LDAP in your container environment, you must create the SSL secret with the certificate of the LDAP server.

   a. Get the root CA that is used to sign your LDAP server and save it to a certificate, for example `ldap-server-cert.crt`. See OpenSSL for instructions to export the root CA of your external service.

   b. To create the secret, run the following command.

      ```
      kubectl create secret generic secretName --from-file=tls.crt=your_cert_path/ldap-server-
      cert.crt
      ```

      Substitute your values for *secretName* and *your_cert_path/ldap-server-cert.crt*. The certificate and key files must be in Privacy Enhanced Mail (PEM) format.

   c. After you obtain the certificate and create the secret, you enable SSL and provide the secret name in the custom resource YAML file in the `ldap_configuration` section.

      ```
      ldap_configuration:
        …
          lc_ldap_ssl_enabled: true
          lc_ldap_ssl_secret_name: "<secretName>"
      ```

      Set the enabled parameter to true and provide your own secret name.

4. Enter the parameter values for your data source instance in the **datasource_configuration** section.

Setting the values for your IBM Content Navigator deployment

The configuration settings for IBM Content Navigator are recorded and stored in the shared YAML file for operator deployment. After you prepare your environment, you add the values for your configuration settings to the YAML so that the operator can deploy your containers to match your environment.

About this task

This topic assumes a fully manual CR YAML creation and edit process. In some cases, your template might be partially or mostly complete before you edit.

The CR YAML file that you use contains elements for a full deployment of the FileNet Content Manager components. Comment out all of the component sections that are not related to your IBM Content Navigator deployment, including the `ecm_configuration` section.

Procedure

1. Open the CR file that you created manually by using the "Checking and completing your custom resource" on page 31.
2. Check the values to make sure they are the values that you want to deploy. If you need more configuration parameters, then use the fully customizable template to copy lines from and paste them into your CR file.

   Consider the following information as you record the values for your deployment environment in the CR file:

   Shared configuration settings

   Un-comment and update the values for the shared configuration, LDAP, datasource, monitoring, and logging parameters, as applicable.

   Remember: Set `shared_configuration.sc_deployment_platform` to a blank value if you are deploying on a non-OpenShift certified Kubernetes platform.

   Use the secrets that you created for the `root_ca_secret` and `trusted_certificate_list` values.

   For more information about the shared parameters, see the following topics:

   - Shared parameters
   - LDAP parameters
   - Datasource parameters

   IBM Content Navigator settings

   Use the `navigator_configuration` section of the custom YAML to provide values for the configuration of IBM Content Navigator. You provide details for configuration settings that you have already created, like the names of your persistent volume claims. You also provide names for pieces of your IBM Content environment, and tuning decisions for your runtime environment.

   You can configure a Navigator desktop to make the **Send Email** context menu option available to end users. Use the `java_mail` settings to configure Send Email for your Navigator deployment. For example:

   ```
   navigator_configuration:
      java_mail:
          host: "my_exchange1.com"
          port: "123"
          sender: "MailAdmin@myexchange.com"
          ssl_enabled: false
   ```

   If you do not want to include Send Mail in your deployment, ensure that the `java_mail` parameter section is commented out.

   For more information about the settings, see IBM Content Navigator parameters.

Validating the YAML in your custom resource file

You must validate your custom resource (CR) file before you apply it. It is likely that you edited the file multiple times, and possibly introduced errors or missed values during your customizations.

About this task

A good check before you apply the custom resource (CR) is to validate the YAML is executable. The web has many tools that you can use, but http://www.yamllint.com is simple and reliable.

Procedure

1. Go to http://www.yamllint.com/.
2. Copy and paste the contents of your CR file into the gray text box, and click **Go**.

Results

The tool reports whether the YAML is valid. If the YAML is valid, it strips out the comments and displays the configuration in a UTF-8 format. Skim the configuration again and check for any values that still show "<Required>". If you kept the comments in your CR, checking it without these lines can make it easier to read.

If the YAML is not valid, the tool generates a message and indicates the line where the error is found.

Applying the custom resource

To install the deployment, you must apply the custom resource to the operator.

Procedure

1. Check that all the capabilities that you want to install are configured.

   ```
   cat ibm_icn_cr_final.yaml
   ```

2. Deploy the configured capabilities by applying the custom resource.

   Using the OpenShift CLI:

   ```
   oc apply -f ibm_icn_cr_final.yaml
   ```

   Using the Kubernetes CLI:

   ```
   kubectl apply -f ibm_icn_cr_final.yaml
   ```

Results

The operator reconciliation loop can take some time. You must verify that the automation containers are running.

1. You can open the operator log to view the progress. Using the OpenShift CLI:

   ```
   oc logs <operator pod name> -c operator -n <project-name>
   ```

   Using the Kubernetes CLI:

   ```
   kubectl logs <operator pod name> -c operator -n <project-name>
   ```

2. Monitor the status of your pods from the command line. Using the OpenShift CLI:

   ```
   oc get pods -w
   ```

   Using the Kubernetes CLI:

```
kubectl get pods -w
```

3. When all of the pods are "Running", you can access the status of your services with the following OCP CLI command.

```
oc status
```

Using the Kubernetes CLI:

```
kubectl status
```

Refer to the “Troubleshooting the operator” on page 37 to access the operator logs.

What to do next

Some capabilities need you to follow post-deployment steps. For more information, see “Completing post-deployment startup tasks” on page 35.

Completing post-deployment startup tasks

After you run the container deployment, you perform additional tasks to configure and start your IBM Content Navigator instance.

Completing extra post-deployment tasks on ROKS

(For Version 20.0.2 only) For most deployments on Red Hat OpenShift Kubernetes Service (ROKS), extra steps are needed to ensure that the environment works correctly.

About this task

To create routes with Transport Layer Security (TLS) termination reencryption on ROKS, you must provide the CA certificate, and the certificate and key for that component. Some components are configured to use routes with "reencrypt", and others are configured with "pass-through". For more information, see https://www.openshift.com/blog/self-serviced-end-to-end-encryption-approaches-for-applications-deployed-in-openshift.

Procedure

1. Get the CA certificate ($DESTINATIONCACERTIFICATE).

```
kubectl get secret -o jsonpath="{ .data.tls\.crt }" "{{ meta.name }}-root-ca"  | base64 -d
```

2. Get a certificate ($CERTIFICATE) and key ($CERT_KEY) for a specific component.

```
kubectl get secret -o jsonpath="{ .data.tls\.crt }" "{{ meta.name }}-XXX-ext-tls-secret" |
base64 -d
kubectl get secret -o jsonpath="{ .data.tls\.key }" "{{ meta.name }}-XXX-ext-tls-secret" |
base64 -d
```

Where XXX is the acronym of the component name and `{{ meta.name }}` can be found by running `kubectl get svc -n <namespace>` to get the current services. For Content Manager for example:

```
kubectl get secret -o jsonpath="{ .data.tls\.crt }" "{{ meta.name }}-fncm-ext-tls-secret" |
base64 -d
kubectl get secret -o jsonpath="{ .data.tls\.key }" "{{ meta.name }}-fncm-ext-tls-secret" |
base64 -d
```

3. Create the routes by using YAML files or the ROKS console with the retrieved certificates and keys.

The following YAML shows an example of a route object where the retrieved certificates and key are replaced with the values from steps 1 and 2.

```
################################################################################################
#
# Licensed Materials - Property of IBM
#
# (C) Copyright IBM Corp. 2019 - 2020. All Rights Reserved.
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
################################################################################################
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: '{{ meta.name }}-<XXX>-route'
  namespace: '{{ meta.namespace }}'
  annotations:
    haproxy.router.openshift.io/balance: roundrobin
    haproxy.router.openshift.io/timeout: 300s
    router.openshift.io/sticky_cookie: -sticky_cookie_annotation
  labels:
    servicename: '{{ meta.name }}-<XXX>-svc'
    app: '{{ meta.name }}'
    app.kubernetes.io/instance: '{{ meta.name }}'
    app.kubernetes.io/managed-by: '{{ meta.name }}'
    app.kubernetes.io/name: '{{ meta.name }}'
    release: "20.0.2"
spec:
  port:
    targetPort: https
  to:
    kind: Service
    name: '{{ meta.name }}-<XXX>-svc'
    weight: 100
  wildcardPolicy: None
  tls:
    insecureEdgeTerminationPolicy: Redirect
    termination: reencrypt
    key: |-
    $CERT_KEY
    certificate: |-
    $CERTIFICATE
    destinationCACertificate: |-
    $DESTINATIONCACERTIFICATE
```

Improving security for session cookies

You can improve security for session cookies by adding httpSession configuration to your `overrides` directory.

About this task

Make this change for both your content repository and IBM Content Navigator deployment.

Procedure

- Add httpSession configuration to your `overrides` directory.

  Create an XML file (for example, `zHTTPsession.xml`) with the following content:

```
<server>
   <httpSession
    cookieName="JSESSIONID"
    cookieSecure="true"
    cookieHttpOnly="true"
    cookiePath="/"
   >
   </httpSession>
</server>
```

  Note: Some features are affected by this setting:

Applets
The cookieHttpOnly="true" setting can cause applets to fail. If you plan to use applets, remove this entry from the XML file. Or you can use the HTML-based solution, such as the HTML step processor.

Additional Navigator features
For information on what features are affected by this setting and possible mitigation, see the following information in the IBM Content Navigator Knowledge Center.

Configuring IBM Content Navigator in a container environment

If you included IBM Content Navigator in your container deployment, you must perform some additional configuration to ensure that the application works with your content services environment.

About this task

After you deploy your IBM Content Navigator container, use the IBM Content Navigator administration console to update settings for the container environment.

Procedure

1. Add the Daeja Viewer license files to the configuration overrides directory for IBM Content Navigator.
   For example, `/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides`.
2. Update the Daeja Viewer log file path, if necessary.
   The default log file is *`install_dir`*`/ibm/viewerconfig/logs/daeja.log`, where *install_dir* is the directory where IBM Content Navigator is installed. Confirm that the path is updated to reflect your container deployment location, for example, `/opt/ibm/viewerconfig/logs/daeja.log`.
3. Update the Sync Services URL:
   a) In the IBM Content Navigator administration tool, click **Sync Services**.
   b) Update the default value for the public service URL.
      The URL must include the server IP address and port of the IBM Content Navigator route in the OpenShift Cloud Platform environment, for example: `http://ICP_IP_Address:30557/sync/notify`.

Troubleshooting the operator

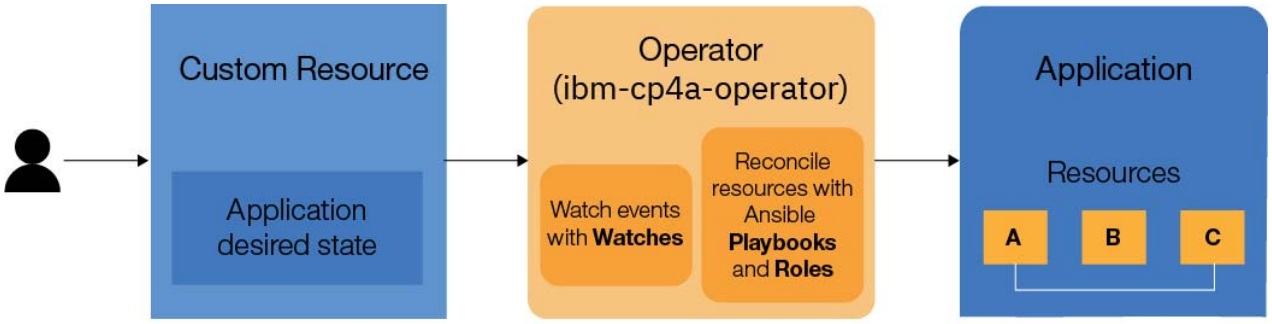
You can use the operator and Ansible containers to retrieve log files of the installed operator.

About this task

The `ibm-fncm-operator` locates the base images and has Ansible roles to handle the reconciliation logic and declare a set of playbook tasks for each component. The roles declare all the variables and defaults for how the role is executed.

The operator deployment creates two containers on your cluster, one for the operator and one for Ansible. By default Ansible uses `/etc/ansible/hosts` as its inventory file; the inventory represents the host machines that Ansible manages. You can create a different inventory file for each project that you have.

The following diagram shows how the operator watches for events, triggers an Ansible role when a custom resource changes, and then reconciles the resources for the deployed applications.



The Ansible container shows the standard Ansible `stdout` logs. To see the logs of a container, run the following command.

```
kubectl logs deployment/ibm-fncm-operator -c ansible > ansible.log
```

For version 20.0.3, run the following command:

```
kubectl logs deployment/ibm-fncm-operator > ansible.log
```

The operator logs contain much more information about the operator than Kubernetes does. To see the logs of the operator container, run the following command.

```
kubectl logs deployment/ibm-fncm-operator -c operator > operator.log
```

For runtime logs, go inside the pod that runs the Ansible container. The runner keeps information about the Ansible run in the container, which is located under `/tmp/ansible-operator/runner/`*`<group>`*`/`*`<version>`*`/`*`<kind>`*`/`*`<namespace>`*`/`*`<name>`*.

If some pods are pending, choose one of the pods and run the following command to get more information.

```
kubectl describe pod <podname>
```

Kubernetes secrets are used extensively, so output about them might also be useful.

```
kubectl get secrets
```

Kubernetes events are objects that provide more insight into what is happening inside a cluster, such as what decisions the scheduler makes or why some pods are evicted from a node. To get information about these events, run the following command.

```
kubectl get events > events.log
```

You can also add the verbose parameter to any **`kubectl`** command.

```
kubectl -v=9 get pods
```

Updating deployments

An update to the custom resource (CR) overwrites the deployed resources. The operator applies the changes during the control loop (observe, analyze, act) that occurs as a result of constantly watching the state of the Kubernetes resources.

Before you begin

If you plan to update the existing deployment of ICN container, such as update the ICN configuration database name for example, ensure that you back up the databases, the files that are stored on the persistent volumes, and any other data that you want to keep.

About this task

You can reconfigure and update the software that is already installed. You can modify the installed software, remove it, and add new components. You must use the same CR YAML file that you deployed with the operator to make the updates. For example, `scripts/generated-cr/ibm_fncm_cr_final.yaml`.

You can also edit the file manually. The following steps go through the manual procedure.

Procedure

1. Review your CR YAML file to make sure it contains all of your intended modifications.

   ```
   cat /scripts/generated-cr/ibm_fncm_cr_final.yaml
   ```

2. Refer to the information in "Checking and completing your custom resource" on page 31 to pick and choose the configuration parameters to update.

   To remove a capability from the deployment, locate the specific **XXX_configuration** section and delete this line along with all of its parameters.

   Note: Some capabilities have parameters under **shared_configuration** or **datasource_configuration**. If you are sure that no other capability uses them, you can remove these parameters too.

3. When you are done with all of the updates you want to make, run the `apply` command to update to the operator.

   Using the OpenShift CLI:

   ```
   oc apply -f <ibm_fncm_my_cr_final.yaml> --overwrite=true
   ```

   The operator reconciliation loop might take several minutes.

   Monitor the status of your pods by using the OpenShift CLI:

   ```
   oc get pods -w
   ```

   Note: You can also use `oc edit FNCMCluster <MY-INSTANCE>` to open the default UNIX visual editor (vi) in the pod.

4. When all of the pods are "Running", you can access the status of your services with the `status` command.

   Using the OpenShift CLI:

   ```
   oc status
   ```

Uninstalling components

Delete the namespace that you used to install to remove all of the deployed components.

Before you begin

If you need to back up your data, make sure that you take the necessary steps to reinstall the deployment. For example, make a copy of the custom resource (CR) that you used in the environment. Make copies of the security definitions that are used to protect the configuration data in the environment. Make copies of the PVs and persistent volume claims (PVC) in the environment.

About this task

Note: To add or remove components to an existing deployment, see "Updating deployments" on page 39.

To uninstall the operator, cluster role, cluster role binding, service account, and the CRD run the following `kubectl` commands:

```
kubectl delete -f <CR.yaml>
kubectl delete -f descriptors/operator.yaml
kubectl delete -f descriptors/role_binding.yaml
kubectl delete -f descriptors/role.yaml
kubectl delete -f descriptors/service_account.yaml
kubectl delete crd fncmclusters.fncm.ibm.com
```

You can also run the `scripts/deleteOperator.sh` script, which includes the same commands.

```
cd container-samples/scripts
./deleteOperator.sh
```

To uninstall the complete deployment, you can delete the namespace by running the following OpenShift CLI command:

```
oc delete project <project-name>
```

What to do next

After you uninstall, you might want to clean up certain files and secrets that you applied to the cluster for specific capabilities.

Administering components in a container environment

In most cases, administering your container environment for content services is the same as administering your on-premises environment. However, some variations exist for container environments.

About this task

Use the information in this section to understand the administration tasks that are different in a container environment. For all other administration tasks, see Administering IBM Content Navigator components.

Starting and stopping components

For container deployments, you can use the deployment scaling capabilities to start and stop your components.

About this task

When you scale your deployments to zero (0), this operation effectively stops the component or application. You scale the deployment back to your original number to restart the component or application. The deployment scaling capability is available from the Kubectl command line.

Procedure

- Scale your deployment from the Kubectl command line:
  a) List your deployments:

```
kubectl describe deployments
```

  b) Stop the deployed container instances:

```
kubectl scale deployment deployment_name --replicas=0
```

  c) Create new instances of the deployed container, for a restart:

```
kubectl scale deployment deployment_name --replicas=number of container instances
```

Monitoring the components in your container environment

You can use your Red Hat OpenShift console to monitor the components in your container environment. You can also use the external logging and monitoring systems that you optionally configured for the container during deployment.

Managing certificates

You must monitor and manage the certificates that are deployed in your network. Administrators need visibility and control over their SSL environments to help them preempt security breaches, outages, and compliance issues.

Providing the root CA certificate

Connections between the components in a container environment are secured by a common root CA certificate. You can decide whether to use the default root CA that is provided by the operator, or provide your own. All the internal server's certificates are signed by the root CA.

About this task

Certificates that are created by the operator are self-signed. If your policy requires certificates that are signed by a recognized certificate manager, you can provide the certificates for the operator to incorporate.

If you want to use your own root CA certificate, obtain or prepare the CA certificate and create a secret for it before you deploy your operator.

When you enter your parameter values in the custom resource YAML file, you provide the name of this secret as the value for the `root_ca_secret` parameter in the shared configuration parameters.

Important: If you choose to use self-signed certificates, certain features of the product might not work as expected because of modern browser restrictions that are related to self-signed certificates. A browser blocks any redirect to a site that uses a certificate that is not signed by a root CA that is trusted by the browser. This can result in access issues for business applications.

Connecting securely with external services

If you want to integrate external services with your container environment, you must exchange Transport Layer Security (TLS) certificates between your component and the external service.

Importing the certificate of an external service

To integrate with an external service, you must first import its Transport Layer Security (TLS) certificate into the operator trust list.

Procedure

If the root certificate authority (CA) key of the external service is not signed by the operator root CA key, provide the TLS certificate of the external service to the component's truststore.

The certificate includes the root CA key and the key of each component. If the external service is not installed by the same custom resource, the root CA key of the service is not signed by the operator root CA key. If the service is installed by the same custom resource, check the documentation of the external service to see whether it uses the same root CA key.

a) Get the root CA that is used to sign your external service and save it to a certificate, for example `external-service-cert.crt`.

   See OpenSSL for instructions to export the root CA of your external service.

b) To create the secret, run the following command in the OpenShift project:

```
kubectl create secret generic secretName --from-file=tls.crt=your_cert_path/external-service-cert.crt
```

Substitute your values for *secretName* and *your_cert_path/external-service-cert.crt*. The certificate and key files must be in Privacy Enhanced Mail (PEM) format.

c) Add the secret to the component's truststore.

Add the secret to the custom resource in the `shared_configuration.trusted_certificate_list` parameter if you want this service to be trusted by all components installed by the operator.

For example:

```
shared_configuration:
  …
  trusted_certificate_list: [adw-tls-secret, baw-tls-secret]
```

This variable is an array and multiple values can be provided by separating them with a comma as shown in the example.

Exporting the operator root CA key and importing it into an external service

After you import the certificate of the external service into the trust list, if the external service needs to access your component, you must extract the operator root CA key and import it to the truststore of the external service.

Procedure

Extract the operator root CA key and import it to the truststore of the external service.

If you are using the default value of `fncm-root-ca` for the operator root CA, use the following command to find the root CA key:

```
oc get secret fncm-root-ca -o template --template='{{ index .data "tls.crt" }}' | base64 --decode > rootCA.crt
```

If you don't know the root CA key for your component, look in the `shared_configuration.root_ca_secret` in the custom resource file.

Tuning the components in your container environment

Use the Kubernetes guidelines for tuning the performance of your container components.

Procedure

- Use the information at the following location to tune your container environment:

  Managing Resources for Containers

Tuning IBM WebSphere Liberty for IBM Content Navigator components

You can tune WebSphere® Liberty to improve container performance. You create XML override files to create these tuning updates.

About this task

The XML files are placed onto the Liberty configuration volume that corresponds to the `configDropins/overrides` folder for WebSphere Liberty. For more information about the volumes and folders, see Creating volumes and folders for deployment .

Tip: Liberty processes the files in the `configDropins/overrides` in alphabetical order. A later configuration overrides an earlier one. As an example, if `configDropins/defaults` contains `a.xml`, `b.xml`, and `c.xml`, the configuration from `c.xml` takes precedence over `b.xml`, and `b.xml` takes

precedence over a `.xml`. Naming the XML file starting with the letter ‘z’ helps ensure the new parameter values are applied by Liberty.

Procedure

- Tune WebSphere Liberty.

  For general tuning tips, consult the WebSphere Application Server Liberty Knowledge Center topic Tuning Liberty.

- Reduce lightweight third-party authentication (LTPA) timeout errors.

  The LTPA timeout value for forwarded credentials between servers parameter setting specifies how long an LTPA token is valid (in minutes). The token contains this expiration time so that any server that receives the token can verify that the token is valid before proceeding further. When the defined amount of time has passed, all user tokens expire regardless of session activity. Consider increasing this value if one or both of the following conditions occur:

  – Batch processes or other operations run uninterrupted for longer than the currently configured LTPA timeout value.
  – A message appears in the Liberty `messages.log` file or is returned to the client session indicating the LTPA token expired.

  To adjust the LTPA expiration parameter value, add the following clause within <server> </server> tags in a pre-existing overrides XML or a new overrides XML. The example shows a 10 minute timeout expressed as seconds:

  ```
  <server>
    <ltpa expiration="600s"/>
  </server>
  ```

- Tune maxPoolSize for your data sources.

  The number of database client connections required for your workload will fluctuate over time (for example: peak logged in users) both in the short term (more users and batch document processing at end of month processing) and long term (more users from rollouts or increased use of system). Best practice is to monitor connection use and track the trends.

  For each pod there will be a number of processes running that will utilize database connections from the available pool of the Liberty connection manager. The size of the connection pool is controlled by the properties in the data source definition XML:

  – minPoolSize - minimum number of available connections in the pool. default 0.
  – maxPoolSize - maximum number of available connections in the pool. default 50 for most operator created data sources.

  To adjust the maxPoolSize parameter value, add the following clause within <server> </server> tags in a pre-existing overrides XML or a new overrides XML. The example shows an adjustment to the maxPoolSize for one of the data sources created by the operator to connect to an object store database. You must include the id and jndiName for the data source where the maxPoolSize will be overridden.

  ```
  <server>
    <dataSource id="DESIGNDS" jndiName="DESIGNDS">
      <connectionManager maxPoolSize="100" />
    </dataSource>
  </server>
  ```

  For more information see Configuring connection pooling for database connections.

  In the example, other required parameters are not shown since only the particular parameter being overridden is included. Samples of complete data source files for each of the FileNet® Content Manager containers can be found in GitHub repository:

  https://github.com/ibm-ecm/container-samples/tree/5.5.5/CPE/configDropins/overrides

Backup and recovery of a container environment

In a container environment, your data is external to the component container. A backup for your data is similar to an on-premises installation. You also back up your configuration information for the environment.

Procedure

- Back up your environment.

  Containers are stateless, so the data that must be backed up in a container environment is external to the container. In a container environment all external mounts to the container are the areas of storage that should be part of a backup set. See the following information for details about the additional areas of storage that must be backed up: "Creating volumes and folders for deployment" on page 14.

  For the rest of your environment, the backup requirements are the same as for an on-premises installation.

  All offline and hot backup guidelines are the same for container deployments.

- As needed, recover your environment from the backups you created.

  Recovery of a container environment is a much easier task because of the separation of the containers from the application data. Recovery is the same as the processes that are documented for on-premises environments.

  You must also recover the external container mounts for deployed containers as documented in the following topic: "Creating volumes and folders for deployment" on page 14.

  Because this data includes container configuration information, make sure to repopulate the data in the appropriate mounts before you bring any containers online again.

Registering and configuring IBM® Content Navigator plug-ins in a container environment

You can use plug-ins to integrate IBM® Content Navigator with other products or to modify the behavior of the web client. In a container environment, the paths to the plug-ins are different than for an on-premises deployment.

Procedure

To register and configure IBM Content Navigator plug-ins:

1. Move the IBM Content Navigator plug-in files to the designated path in the IBM Content Navigator icn-icp-pluginstore volume, for example: `/icnpluginstore/plugins`
2. Open the administration tool in the web client.
3. Click **Plug-ins** and then click **New Plug-in**.
4. Specify the plug-in file that you want to register.

   The plug-in file must be in the `/opt/ibm/plugins` path in the IBM Content Navigator container volume.
5. Load the plug-in.
6. Provide any additional configuration settings that the plug-in requires.
7. Save your changes.

Configuration reference

When you prepare your environment for container deployment and plan for the setup of your container applications, you collect relevant configuration values for the environment. Use the following reference topics to collect and understand the configuration values that you supply for the container deployment process.

Configuration reference for operators

When you edit the configuration YAML file for deployment, you supply values about your supporting environment and your component configuration. Collect the configuration parameters as you set up your environment for the content services container deployment.

Shared parameters

Update the custom YAML file to provide the details of an IBM Security Directory server or Microsoft Active Directory LDAP server.

Table 2. Spec configuration parameters

Parameter	Description
appVersion	The version of the current release.

Table 3. Shared configuration parameters

Parameters	Description	Default Values
sc_deployment_context	Do not change this default setting.	FNCM
image_pull_secrets	Shared image pull secrets.	[]
sc_image_repository	All components must use the same Docker image repository, co.icr.io for entitled register, or a local Docker image repository.	cp.icr.io
root_ca_secret	If you provide your own root certificate, enter the value.	fncm-root-ca
sc_deployment_type	Set the value to "demo" for an evaluation deployment, "non-production" or "production". Set the value to `demo` for an evaluation deployment, and `enterprise` for CM8, CMOD, and P8. **Note:** If you set the value to "demo", Db2® Universal Container and OpenLDAP instances are created as part of the installation. If the value is set to "demo", you must also set the **`sc_dynamic_storage_classname`** parameter.	demo

Table 3. Shared configuration parameters (continued)

Parameters	Description	Default Values
sc_run_as_user	Optionally specify a RunAs user for the security of the pod. This is usually a numerical ID.	
sc_deployment_platform	Enter your certified Kubernetes platform type	OCP
sc_deployment_hostname_suffix	Specify the hostname used to create routes. Routes are created automatically if **sc_deployment_platform** is set to OCP or ROKS.	example.com
trusted_certificate_list	If you want to use your own certificate, use the certificate file to create a secret and then add the secret for this parameter.	[]
encryption_key_secret	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Shared encryption key secret name that is used for Workstream Services and Process Federation Server integration.	icp4a-shared-encryption-key
shared_configuration.sc_deployment_patterns	For CM8 and CMOD, remove the default value and set to blank.	content
sc_optional_components	(For CM8 and CMOD, remove the default value and set to blank.) A component is identified by an acronym of the component or subcomponent name. If you want to install more than one component, separate the acronyms of the components with a comma, for example, css,es,tm.	cmis
sc_content_initialization	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Enable/disable `content` initialization (creation of P8 domain, creation of object stores, creation of CSS servers, and initialization of Navigator (ICN)). If the "**initialize_configuration** " section is defined in the CR, then that configuration takes precedence over this parameter.	false

Table 3. Shared configuration parameters (continued)		
Parameters	**Description**	**Default Values**
sc_content_verification	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Enable/disable the `content` verification (creation of test folder, creation of test document, execution of CBR search, and creation of Navigator demo repository and desktop). If the "**`verify_configuration`**" section is defined in the CR, then that configuration takes precedence over this parameter.	false
sc_fast_file_storage_classname sc_medium_file_storage_classname sc_slow_file_storage_classname	Storage for an enterprise deployment needs 3 storage classes. If you do not have 3 storage classes, you can use the same one for "slow", "medium", and "fast". On OpenShift Container Platform (OCP) 3.x and 4.x, the deployment script sets these parameters based on the user input.	cp4a-file-retain-gold-gid cp4a-file-retain-silver-gid cp4a-file-retain-bronze-gid

LDAP parameters

Update the custom YAML file to provide the details that are relevant for your FileNet Content Manager and IBM Content Navigator LDAP environment. Parameters marked with (External users) apply only for environments that are using the 2-LDAP method for supporting External Share.

Table 4. LDAP configuration parameters		
Parameters	**Description**	**Default Values**
LDAP server type lc_selected_ldap_type:	The type of the directory service provider you are using for your container environment. Choices are IBM Security Directory Server or Microsoft Active Directory	IBM Security Directory Server
LDAP server host name lc_ldap_server	The host name for the LDAP server that you are using for the environment.	<hostname>
LDAP port lc_ldap_port	The port number for the LDAP server that you are using.	389
Secret lc_bind_secret	User name and password for the bind user. The LDAP bind secret must have ldapUsername and ldapPassword keys.	ldap_bind_secret

Table 4. LDAP configuration parameters (continued)		
Parameters	**Description**	**Default Values**
Base entry distinguished name c_ldap_base_dn	The base distinguished name (DN) of an LDAP user who is allowed to search the LDAP directory if the LDAP server does not allow anonymous access.	dc=hqpsidcdom,dc=com
SSL enablement lc_ldap_ssl_enabled	Specify whether SSL is enabled.	false
SSL secret lc_ldap_ssl_secret_name	Provide the name of the SSL secret that you created.	
User name attribute lc_ldap_user_name_attribute	Provide the format of the user name.	*:cn
User display name attribute lc_ldap_user_display_name_attr	Provide the format of the display name.	cn
Base entry group distinguished name lc_ldap_group_base_dn	The base DN subtree that is used when searching for group entries on the LDAP server.	dc=hqpsidcdom,dc=com
Group name lc_ldap_group_name_attribute	Provide the format of the group name.	*:cn
Group display name lc_ldap_group_display_name_attr	Provide the format of the group display name.	cn
Group membership search filter lc_ldap_group_membership_search_filter	Filter for finding entries in the LDAP base DN (groups) subtree that match the group name.	(|(&(objectclass=groupofnames)(member={0}))(&(objectclass=groupofuniquenames)(uniquemember={0})))
Directory service server group id map lc_ldap_group_member_id_map	The group id is a filter that is used to determine the group name.	groupofnames:member
Max search results lc_ldap_max_search_results	Maximum number of search results to return.	4500
(Active Directory) lc_ad_gc_host	Active Directory host.	
(Active Directory) lc_ad_gc_port	Active Directory port.	

Table 4. LDAP configuration parameters (continued)

Parameters	Description	Default Values
(Active Directory) User filter lc_user_filter	Active Directory user filter.	(&(cn=%v)(objectclass=person))
(Active Directory) Group filter lc_group_filter	Active Directory group filter.	(&(cn=%v) (|(objectclass=groupofnames) (objectclass=groupofuniquenames) (objectclass=groupofurls)))
(IBM Security) User filter lc_user_filter	IBM Security user filter	(&(cn=%v)(objectclass=person))
(IBM Security) Group filter lc_group_filter	IBM Security group filter.	(&(cn=%v) (|(objectclass=groupofnames) (objectclass=groupofuniquenames) (objectclass=groupofurls)))
(External users) LDAP server type lc_selected_ldap_type:	(This parameter does not apply to CM8 and CMOD. Leave the default value.) The type of the directory service provider you are using for your container environment. Choices are IBM Security Directory Server or Microsoft Active Directory	IBM Security Directory Server
(External users) LDAP server host name lc_ldap_server	(This parameter does not apply to CM8 and CMOD. Leave the default value.) The host name for the LDAP server that you are using for the environment.	<hostname>
(External users) LDAP port lc_ldap_port	(This parameter does not apply to CM8 and CMOD. Leave the default value.) The port number for the LDAP server that you are using.	389
(External users) Base entry distinguished name c_ldap_base_dn	(This parameter does not apply to CM8 and CMOD. Leave the default value.) The base distinguished name (DN) of an LDAP user who is allowed to search the LDAP directory if the LDAP server does not allow anonymous access.	dc=hqpsidcdom,dc=com
(External users) SSL enablement lc_ldap_ssl_enabled	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Specify whether SSL is enabled.	false

Table 4. LDAP configuration parameters (continued)		
Parameters	**Description**	**Default Values**
(External users) SSL secret lc_ldap_ssl_secret_name	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Provide the name of the SSL secret that you created.	
(External users) User name attribute lc_ldap_user_name_attribute	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Provide the format of the user name.	*:cn
(External users) User display name attribute lc_ldap_user_display_name_attr	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Provide the format of the display name.	cn
(External users) Base entry group distinguished name lc_ldap_group_base_dn	(This parameter does not apply to CM8 and CMOD. Leave the default value.) The base DN subtree that is used when searching for group entries on the LDAP server.	dc=hqpsidcdom,dc=com
(External users) Group name lc_ldap_group_name_attribute	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Provide the format of the group name.	*:cn
(External users) Group display name lc_ldap_group_display_name_attr	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Provide the format of the group display name.	cn
(External users) Group membership search filter lc_ldap_group_membership_search_filter	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Filter for finding entries in the LDAP base DN (groups) subtree that match the group name.	(|(&(objectclass=groupofnames)(member={0}))(&(objectclass=groupofuniquenames)(uniquemember={0})))
(External users) Directory service server group id map lc_ldap_group_member_id_map	(This parameter does not apply to CM8 and CMOD. Leave the default value.) The group id is a filter that is used to determine the group name.	groupofnames:member
(External users) Max search results lc_ldap_max_search_results	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Maximum number of search results to return.	4500
(External users) (Active Directory) lc_ad_gc_host	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Active Directory host.	

Table 4. LDAP configuration parameters (continued)		
Parameters	**Description**	**Default Values**
(External users) (Active Directory) lc_ad_gc_port	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Active Directory port.	
(External users) (Active Directory) User filter lc_user_filter	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Active Directory user filter.	(&(cn=%v)(objectclass=person))
(External users) (Active Directory) Group filter lc_group_filter	(This parameter does not apply to CM8 and CMOD. Leave the default value.) Active Directory group filter.	(&(cn=%v) (|(objectclass=groupofnames) (objectclass=groupofuniquenames) (objectclass=groupofurls)))
(External users) (IBM Security) User filter lc_user_filter	(This parameter does not apply to CM8 and CMOD. Leave the default value.) IBM Security user filter	(&(cn=%v)(objectclass=person))
(External users) (IBM Security) Group filter lc_group_filter	(This parameter does not apply to CM8 and CMOD. Leave the default value.) IBM Security group filter.	(&(cn=%v) (|(objectclass=groupofnames) (objectclass=groupofuniquenames) (objectclass=groupofurls)))

Datasource parameters

Update the custom YAML file with details of your ICN configuration database. Some parameters are specific to database vendors.

Table 5. Datasource configuration parameters		
Parameters	**Description**	**Default Values**
(ICN) Database type dc_database_type	Specify the type for your IBM Content Navigator database. Possible values are db2, db2HADR, oracle, sqlserver, or postgresql.	db2
(ICN) Database URL dc_oracle_icn_jdbc_url	Oracle: Provide the URL for the IBM Content Navigator database.	jdbc:oracle:thin:@// <hostname>:1521/orcl
(ICN) Datasource name dc_common_icn_datasource_name	The JNDI name of the non-XA JDBC data source associated with the IBM Content Navigator table space or database. The name must be unique.	ICMClientDS
(ICN) Database server host database_servername	The host name of the server where the database software is installed.	<hostname>

Table 5. Datasource configuration parameters (continued)		
Parameters	**Description**	**Default Values**
(ICN) Database port database_port	Provide the database port.	50000
(ICN) Database name database_name	Provide the database name.	ICNDB
(ICN) Standby server name dc_hadr_standby_servername	Enter the standby server name.	<hostname>
(ICN) Standby server port dc_hadr_standby_port	Enter the standby server port.	50000
(ICN) Validation timeout dc_hadr_validation_timeout	Specify the validation timeout entry.	3
(ICN) Retry interval for client reroute dc_hadr_retry_interval_for_client_reroute	Specify the time in seconds between connection attempts made by the automatic client reroute if the primary connection to the server fails.	3
(ICN) Max retries for client reroute dc_hadr_max_retries_for_client_reroute	The maximum number of connection retries attempted by automatic client reroute if the primary connection to the server fails. This property is used only if the Retry interval for client reroute property is set.	3

IBM Content Navigator parameters

Update the custom YAML file to provide the details that are relevant to your IBM Content Navigator and your decisions for the deployment of the container.

Table 6. Configuration parameters		
Parameters	**Description**	**Default Values**
ban_secret_name	Contains the information about the LDAP user and password for components.	"{{ meta.name }}-ban-ext-tls-secret"
ban_ext_tls_secret_name	If you create a tls secret, use this parameter to specify it for IBM Content Navigator. Otherwise the operator creates one for you.	"{{ meta.name }}-ban-ext-tls-secret"
ban_auth_ca_secret_name	If you create a ca secret, use this parameter to specify it for IBM Content Navigator. Otherwise the operator creates one for you.	"{{ meta.name }}-ban-auth-ca-secret"
Replica count replica_count	How many Content Platform Engine replicas to deploy.	2

Table 6. Configuration parameters (continued)

Parameters	Description	Default Values
Run as user run_as_user	User to run the deployment. If run_as_user is commented out, the deployment uses a UID auto-assigned by Open Shift.	
Image details repository tag pull_policy	Specifies the image to be used.	co.cir.io/cp/cp4a/fncm/navigator-sso ga-308-icn Always
Logging for workloads (log) format:	The format for workload logging.	json
Resources -> requests cpu	Specifies a CPU request for the container.	500m
Resource -> requests memory	Specify a memory request for the container.	512Mi
Resource -> limits cpu	Specify a CPU limit for the container.	1
Resource -> limits memory	Specify a memory limit for the container.	1536Mi
Auto Scale enabled	Specify whether to enable auto scaling.	false
Auto Scale max_replicas	The upper limit for the number of pods that can be set by the autoscaler. Required.	3
Auto Scale min_replicas	The lower limit for the number of pods that can be set by the autoscaler. If it is not specified or negative, the server will apply a default value.	1
Auto Scale target_cpu_utilization_percentage	The target average CPU utilization (represented as a percent of requested CPU) over all the pods. If it is not specified or negative, a default autoscaling policy is used.	80
java_mail.host	Specify the host of the mail session.	fncm-exchange1.ibm.com
java_mail.port	Specify the port to use with the mail session host.	25

Table 6. Configuration parameters (continued)		
Parameters	**Description**	**Default Values**
java_mail.sender	For sender, enter a user that has access to the email server to log on.	MailAdmin@fncmexchange.com
java_mail.ssl_enabled	Specify whether SSL is enabled.	false
Route public hostname hostname	Provide a hostname that the operator uses to create an OpenShift® route definition to access the application, most often the host name of the infrastructure node in Open Shift. This parameter does not apply for non-OpenShift platforms.	<hostname>
Time Zone for container (TZ)	The time zone for the container deployment.	Etc/UTC
Initial percentage jvm_initial_heap_percentage	The initial use of available memory.	18%
Maximum percentage jvm_max_heap_percentage	The maximum percentage of available memory to use.	33%
Custom JVM arguments jvm_customize_options	Optionally specify JVM arguments using comma separation. For example: jvm_customize_options="-Dmy.test.jvm.arg1=123,-Dmy.test.jvm.arg2=abc,-XX:+SomeJVMSettings,XshowSettings:vm" If needed, you can use DELIM to change the character that is used to separate multiple JVM arguments. In this example, a semi-colon is used to separate the JVM arguments: jvm_customize_options="DELIM=;-Dcom.filenet.authentication.wsi.AutoDetectAuthToken=true;-Dcom.filenet.authentication.providers=ExShareUmsInternal,ExShareIbmId,ExShareGID"	None
Navigator JNDI datasource name icn_jndids_name	Name for the Navigator JNDI datasource.	ECMClientDS
Schema icn_schema	Schema for IBM Content Navigator.	ICNDB

Table 6. Configuration parameters (continued)

Parameters	Description	Default Values
Table space icn_table_space:	Table space for IBM Content Navigator.	ICNDB
Administrator icn_admin	IBM Content Navigator administrator user.	CEADMIN
Accept license license	The value must be set to accept to deploy.	accept
enable_appcues	Internal use only. Do not change the value.	false
allow_remote_plugins_via_http	It is recommended not to change this setting.	true
Enable monitoring monitor_enabled	Specify whether to use the built-in monitoring capability.	false
Enable logging logging_enabled	Specify whether to use the built-in logging capability.	false
Enable Graphite collectd_enable_plugin_write_graphite	If you use Graphite database for metrics or use IBM Cloud® monitoring, set to true.	false
Configuration overrides PVC name (existing _pvc_for_icn_cfgstore)	The persistent volume claim for IBM Content Navigator configuration.	icn-cfgstore
Logs PVC name (existing _pvc_for_icn_logstore)	The persistent volume claim for IBM Content Navigator logs.	icn-logstore
Plug-in PVC name (existing _pvc_for_icn_pluginstore)	The persistent volume claim for the plug-ins.	icn-pluginstore
Viewer cache PVC name (existing _pvc_for_icnvw_cachestore)	The persistent volume claim for the viewer cache.	icn-vw-cachestore
Viewer log PVC name (existing _pvc_for_icnvw_logstore)	The persistent volume claim for the viewer log.	icn-vw-logstore
Aspera (existing _pvc_for_icn_aspera)	The persistent volume claim for Aspera.	icn-asperastore

Table 6. Configuration parameters (continued)		
Parameters	**Description**	**Default Values**
probe > readiness initial_delay_seconds period_seconds timeout_seconds failure_threshold	The behavior of readiness probes to know when the containers are ready to start accepting traffic.	120 5 10 6
probe > liveness initial_delay_seconds period_seconds timeout_seconds failure_threshold	The behavior of liveness probes to know when to restart a container.	600 5 5 6
Image pull secrets name	The secrets to be able to pull images.	admin.registrykey