IBM FileNet P8 Platform
Version 5 Release 5

Using Containers in V5.5.1 to V5.5.4

IBM

Contents

Chapter 1. Overview of the content services containers

Starting in V5.5.1, containers provide a way to deploy a full FileNet® P8 environment in a fraction of the time required for a standard on-premises installation. Container deployments also enable you to patch or upgrade FileNet P8 components more quickly.

Important: Content services containers are available in V5.5.1 and later.

You can deploy content services containers on an IBM Cloud® Private environment or on a Kubernetes environment with a Docker server.

Deploying on a container platform provides the following benefits:

- Rapid deployment of FileNet P8 Platform components
- Improved patching and upgrading for components
- Dynamic scalability when running on the Kubernetes container platform
- Improved resiliency when running on the Kubernetes container platform

Available containers

The following components are available as a container:

- Content Platform Engine
- Content Search Services
- IBM® Content Navigator
- Content Management Interoperability Services (CMIS)
- (V5.5.2 and later) IBM FileNet External Share
- (V5.5.4 and later) Content Services GraphQL API

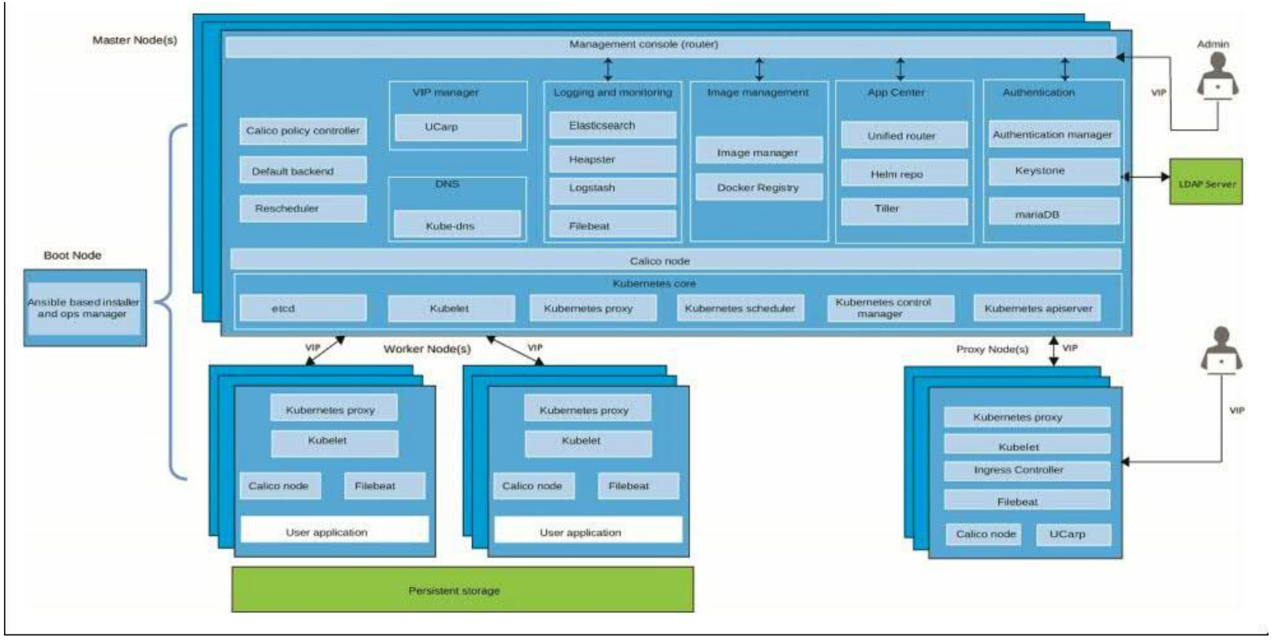
For deployments on IBM Cloud Private, the IBM Business Automation Configuration Container is also offered. When deployed, this container provides a configuration tool that offers a more streamlined configuration experience than the other container deployment methods.

Containers on IBM Cloud Private

IBM Cloud Private is an application platform for developing and managing on-premises, containerized applications. It is an integrated environment for managing containers that includes the container orchestrator Kubernetes, a private image registry, a management console, and monitoring frameworks.

You download the Passport Advantage® packages for the content services components and the configuration container, create an archive file of images, and import the archive into the IBM Cloud Private catalog. This makes the content services containers available as catalog items in IBM Cloud Private. You can use the IBM Business Automation Configuration Container to deploy the content services components in the IBM Cloud Private environment.

The following graphic depicts the architecture for an IBM Cloud Private container environment:



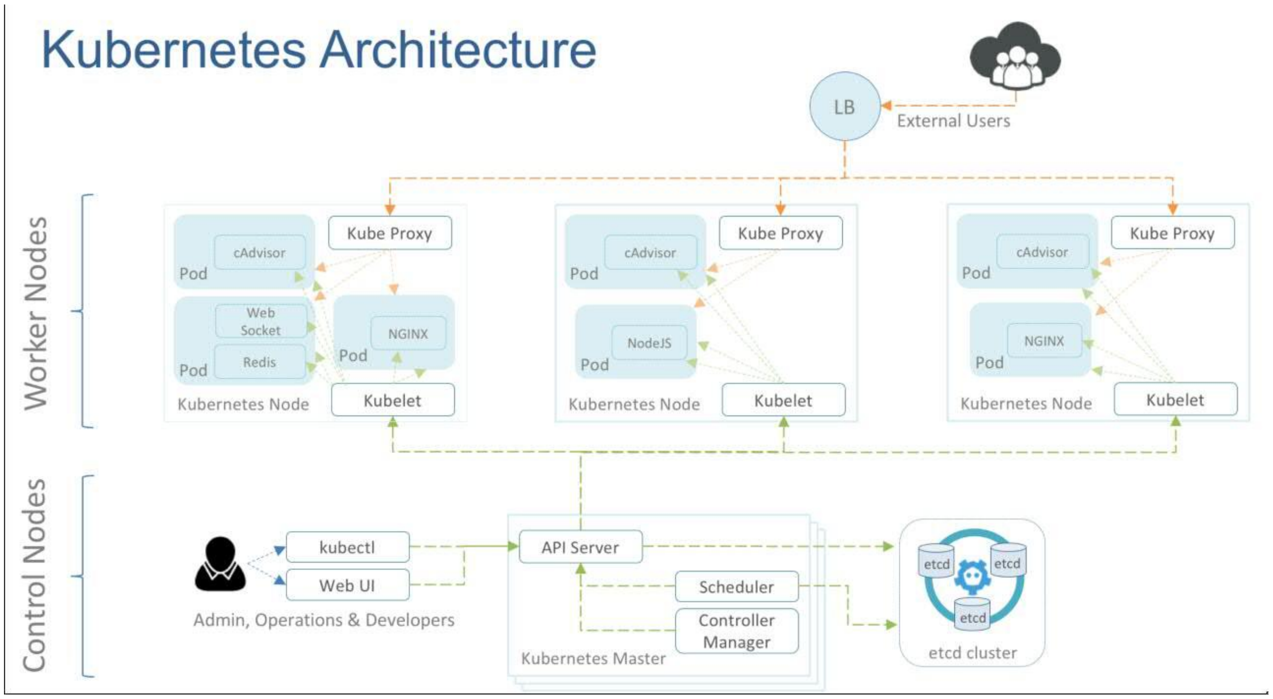
Containers on Kubernetes

Kubernetes is an open-source platform for managing containerized workloads. The Kubernetes platform provides ease of deployment and management of containerized applications like FileNet P8 Platform. Additionally, it provides out-of-the-box capabilities to enhance the runtime environment of FileNet P8 Platform components. You can provide improved resiliency, dynamic scalability, and load balancing for FileNet P8 Platform components by using the native Kubernetes command line interface (kubectl).

You use Kubernetes deployments to place instances of applications, which are built into Helm charts that reference Docker images. The Helm charts contain the details about your application, and the Docker images contain all the software packages that your applications need to run.

Important: Deploying content services containers is tested and supported on Kubernetes that is provided by IBM Cloud Private. Contact IBM if you want to deploy content services containers on a different certified Kubernetes provider.

The following graphic depicts the architecture for a Kubernetes container environment:



When to choose containers

Deploying FileNet P8 containers instead of an on-premises installation can be preferable in a number of possible scenarios. But there are also reasons to maintain a standard on-premises installation model. Consider the relevant factors when deciding whether to use containers.

Deploying products in a container environment can be faster and easier than a standard on-premises installation. Additionally, updating software in a container environment can be simpler and less disruptive.

If you have the following requirements for your content services environment, you might want to choose a container deployment:

- Dynamic scalability to adjust to workload
- Zero downtime support for patching
- Ability to not only quickly patch or upgrade but quickly rollback to previous versions
- Better resource usage and management
- Better resiliency for your applications

If you have any of the following requirements, you might want to choose or maintain a standard on-premises installation:

- Your platform and software choices are currently not supported by the container platform. For more information, see the Software and Product Compatibility Report.
- You use custom applications that use the Content Platform Engine EJB protocol.
- You use applications integrated with IBM Content Navigator, such as IBM Enterprise Records, that are not available for container deployment.
- You use a single IBM Content Navigator instance to connect to Content Manager on Demand and IBM Content Manager in addition to IBM FileNet Content Manager. Currently, only FileNet P8 Platform repositories are supported in the container environment.
- You use Content Platform Engine Virtual Member Manager directory configuration
- You use the IBM Content Navigator Task Manager features, for example, Teamspace deletion or Box share

- You use the IBM Spectrum Protect fixed content device for Content Platform Engine storage

Known issues and limitations for content services containers

The following information describes known issues and limitations when you deploy content services containers. These limitations cover the deployment process as well as using the content services components in a container environment.

Support details for content services components in container environments

For details about support in container environments, see the Software Product Compatibility report for the following products:

- FileNet Content Manager V5.5.1, V5.5.2, V5.5.3, V5.5.4
- IBM Content Foundation Container V5.5.1, V5.5.2, V5.5.3, V5.5.4
- IBM Content Navigator Container V3.0.4, V3.0.5, V3.0.6, V3.0.7

Limitations for IBM Content Navigator

(V5.5.3 and earlier) IBM Content Navigator Task Manager

The IBM Content Navigator Task Manager feature and any functions associated with the feature are not supported when IBM Content Navigator is deployed in a container environment. This limitation includes the following functions:

- Teamspace deletion
- IBM Content Navigator Box Share
- IBM Case Manager Box Integration
- IBM Enterprise Records sweep
- IBM Content Collector for SAP

Limitations for Content Platform Engine

EJB transport and any client applications or features that require EJB transported are not supported when Content Platform Engine is deployed in a container environment.

(V5.5.4) Known Issue: zero-byte file causes error message

When you deploy the external share container with the operator, but do not use the external LDAP settings, a zero-byte file called `ibm_ext_ldap_AD.xml` is created inside the container. This file triggers an invalid file error message. To solve the issue, you must manually delete the file from the disk.

(V5.5.1) Known Issue: The Initialization and Verification options in the configuration tool are not working properly for environments that are using Oracle databases

If you are using Oracle database configured through the IBM Business Automation Configuration Container, do not select the optional steps Initialize and Verification in the configuration tool. If a user uses Initialize and Verify with an Oracle database the object stores are not created correctly.

By default the IBM Business Automation Configuration Container Initialize option configures shared database connections for the object store. However, this process works only with certain prerequisites in place. The recommended approach is to not use the Initialize and Verify options in the configuration tool so that you don't encounter issues due to the shared database connections without the proper prerequisites.

Workaround:

If you are deploying with an Oracle database, make sure that the following options in the configuration tool are not selected:

- Include initialization step
- Include verification step

When your deployment completes, you must initialize your Content Platform Engine and other components manually. See the following topic for details: "Initializing and verifying your content services environment" on page 112

If you selected the options during configuration, you can fix the environment by deleting the created object store, cleaning the associated tablespace, and recreating the object stores manually through the Administration Console for Content Platform Engine.

(V5.5.3 and earlier) Known Issue: When using IE or Edge browser to access the IBM Business Automation Configuration Container tool, the log output might not display as expected

On the deployment result page, the job container displays the log output in real time. When using an Internet Explorer or Edge browser, sometimes no log output displays and the deployment progress continues to show as not started.

The EventSource API that is used to fetch the server log information is not supported by Internet Explorer and Edge browsers, as explained in the following information: https://developer.mozilla.org/en-US/docs/Web/API/EventSource#Browser_compatibility

Workaround:

Use other browsers like Chrome, Safari, and Firefox.

Known Issue: In some cases the rerun function displays the log of the previous event before starting with the new event information

When a job fails, after you troubleshoot and fix the issues, you can rerun the failed job by clicking **Rerun** in the user interface.

When you click Rerun, it can take some time for the backend job container to restart and clean the log of the previous run. It can happen that the log is requested before the previous log is cleaned, and in this situation it prints the older log.

Resolution:

To get a clean view, after the new log starts to display messages, refresh the page. The old log is removed from the display.

(V5.5.1) Known Issue: Initializing IBM Content Navigator

When you choose the **Include initialization step** while deploying IBM Content Navigator, the application initializes a repository and a new default desktop.

Known Issue: When you try to edit the repository, the **Save** button is disabled.

Reason: The repository cannot be saved because the default repository ID includes an underscore ("_").

Impact: You cannot modify the repository settings. However, there is no impact to the functionality connecting to the object store in FileNetP8. Users still can browse, create documents, and add documents to the repository.

Resolution: Update the following fields in the `jobs.yml` file to values that do not include an underscore:

- ADD_REPO_ID
- ADD_DESKTOP_REPO_ID

Chapter 2. Planning for container deployment

Depending on your starting point, your actions to prepare for and deploy your container environment can vary. Consider the scenario that best matches your goals for the container deployment.

The preparation steps and deployment process are different depending on both your starting point and your goal for the container environment. Although the component applications are deployed in the container environment, in most cases the environment still requires many of the same FileNet P8 infrastructure components and preparation steps that are needed in an on-premises installation.

Review the roadmaps for the following deployment scenarios to determine the steps you need to take to prepare for and run the container deployment:

Creating an all-in-one developer environment
This scenario works well for a demonstration server or non-production developer workstation. Because this complete platform relies on open LDAP and a generic database configuration, this installation method is not appropriate for production-level use. You can use the all-in-one installation tool to create environments for demonstration or testing purposes, or to try out the FileNet P8 system in a container environment before you move to containers.

Creating a new instance of FileNet P8 with container deployment
This scenario assumes that you are creating an entirely new environment for FileNet P8. To prepare for this environment, you must follow the same planning and preparing topics as you would for an on-premises installation, with some exceptions. For example, you must create a database, configure your LDAP system, create users, and take other steps to prepare before you are ready to deploy the component containers.

Adding container instances to an existing FileNet P8 domain
In this scenario, you already have an on-premises installation of FileNet P8 running. You are either adding container instances of the applications alongside your existing system, or moving from your existing on-premises system to one in which your components are deployed as containers.

Because you already have a working FileNet P8 environment, you do not need to set up your database, LDAP server, users, and so on. However, you must still take steps to prepare your environment and the container environment for deployment.

Roadmap for a new FileNet P8 domain in a container environment

You prepare your FileNet® P8 infrastructure components as well as the servers and configuration settings that are required for the container environment, then use the method you choose to deploy the component containers.

FileNet® P8 requires a robust environment of integrated software, such as databases, Lightweight Directory Access Protocol (LDAP) servers for access configuration, containers, and multiple required or optional server and client components in the FileNet P8 family. This interdependence calls for preparation and collaboration with the administrators in your environment to configure the infrastructure and record the relevant details for the FileNet P8 deployment and setup.

The following roadmap presents a version of the preparation and deployment steps for your FileNet P8 setup process:

1. Read the whole Planning and Preparing for FileNet P8 section, and make the following decisions:

   - Which administrators and team members in your organization to designate for the different administrative roles and tasks
   - Which installation scenario is most relevant and useful for your needs

   Note: Because a web application server is included in the component container deployment, preparation steps for application server settings do not apply.

2. Complete the tasks to prepare for deployment. These tasks are likely to be performed by several different administrators. Create a collaborative record to gather all the settings you need for the IBM

Business Automation Content Services Configuration tool or YAML file that you run to perform the deployment.

- Configure storage for the FileNet P8 domain.
- Set up the directory server accounts for all of the roles and users your environment requires.
- Create databases to contain the Content Platform Engine GCD and object stores. You can create and deploy up to 2 object stores as part of the container deployment process.
- If you plan to include the IBM Content Navigator container, do the following additional steps:
  - Set up the directory server accounts for all of the roles and users your IBM Content Navigator environment requires.
  - Create a database to contain the IBM Content Navigator configuration information.
- Prepare the server instance that will host the content services applications. For example, for IBM Cloud Private you might create a distinct user and namespace to facilitate the management of the content services deployed components.
- Set up the worker nodes where the component containers will be deployed. This includes preparation of shared storage for configuration information, logs, and the document content and optional full text indexes, along with associated persistent volumes and persistent volume claims.

3. Make the component container images available on the IBM Cloud Private or Kubernetes server.
4. Use the settings recorded by your setup administrators to complete the configuration tool fields or YAML file, and run the deployment. This step deploys your applications and prepares them for use.

Roadmap for a container deployment with an existing FileNet P8 domain

When you deploy containers to work with an existing FileNet P8 domain, your environment infrastructure is already set up. You prepare the environment by upgrading to the appropriate version.

Upgrade planning depends on the details of your existing environment. The starting version and platform choices all influence the upgrade path of your existing components.

It is recommended to first upgrade the WebSphere Application Server on-premises system to the targeted level of FileNet Content Manager. Run the FileNetCPE Engine with its associated client applications to ensure the system is operating as expected. Then move the FileNet P8 domain to be managed by a container based deployment.

However it is supported to both upgrade and move at the same time.

To upgrade to Content Platform Engine V5.5.x, your system must be at V5.2.0 or later. If you want to move your environment to a deployment on a Certified Kubernetes platform, your FileNet P8 domain must be at V5.5.1 or later. Use the following information to complete that initial upgrade step: Planning and preparing for FileNet P8 upgrade.

The following roadmap presents a version of the preparation and deployment steps for your FileNet P8 setup process:

1. Read the Upgrade scenarios section, and make the following decisions:
   - Will the current WebSphere® Application Server hosted deployment of the FileNetEngine application continue to be used in a side-by-side configuration with the Content Platform Engine container deployment?
     - If yes, then the WebSphere Application Server hosted deployment must be upgraded to the same version of FileNet Content Manager as the container-hosted deployment. Plan to upgrade the WebSphere Application Server hosted environment first, to Content Platform Engine V5.5.2 or later. After the upgrade, use the following information to add containers to an existing P8 Domain that is managed by WebSphere Application Server: "Adding containers to an existing on-premises P8 domain (side-by-side deployment)" on page 119
     - If no, then plan to shut down the existing WebSphere Application Server hosted environment and use the following information to upgrade an existing P8 Domain using containers only: "Upgrading an existing P8 domain to a containers-only environment" on page 120

Note: All data in the existing global configuration database, object stores, and workflow system are automatically upgraded when you deploy the Content Platform Engine container.

- Will it be necessary to first upgrade or migrate any underlying vendor software supported by Content Platform Engine? See the following topic for more information: Upgrading or migrating the underlying vendor software supported by Content Platform Engine.

Note: Because a web application server is included in the component container deployment, preparation steps for application server settings do not apply.

2. Complete the tasks to prepare for deployment. These tasks are likely to be performed by several different administrators. Create a collaborative record to gather all the settings you need for the IBM Business Automation Content Services Configuration tool or YAML file that you run to perform the deployment.
   - Ensure that the directory server is supported by the new Content Platform Engine version.
   - Prepare the database server that contains the Content Platform Engine GCD and object stores for upgrade.
   - If you plan to include the IBM Content Navigator container, prepare the database that contains the IBM Content Navigator configuration information for upgrade.
   - Prepare the server instance that will host the content services applications. For example, for IBM Cloud Private you might create a distinct user and namespace to facilitate the management of the content services deployed components.
   - Set up the worker nodes where the component containers will be deployed. This includes preparation of shared storage for configuration information, logs, and the document content and optional full text indexes, along with associated persistent volumes and persistent volume claims.
3. Make the component container images available on the IBM Cloud Private or Kubernetes server.
4. Use the settings recorded by your setup administrators to complete the configuration tool fields or YAML file, and run the deployment. This step deploys your applications and prepares them for use.

(Optional) Upgrade with migration to containers using a staging environment

An upgrade can be accomplished while also migrating from an on-premises environment to a container environment. Making such a change is often part of the motivation for doing the upgrade and it is important to have a well-understood process.

This topic applies only if you want to use a replica or staging environment during the upgrade to minimize the impact of the upgrade on the production system.

Upgrading large FileNet P8 systems involves significant work. The upgrade can be particularly challenging if you are changing the underlying platform of major system components, such as Content Platform Engine. Using this approach, you might install and configure a new server instance, such as for the database server. The initial installation and configuration work can be done without impacting the production system.

At a high level, complete the upgrade migration procedures by using the following steps. Some steps are repeated for each major FileNet P8 component:

- Determine a time when you can run the upgrade, which must be done when nobody is altering the production system data. The copy of the production data (replica) must reflect the production system. Otherwise the upgrade is not on current data.
- Set up a second system that contains a copy of production data. With this approach, you can revert to the original system if you encounter problems during the upgrade. You can also do some of the initial installation and configuration without impact to the production system. This second system lets you move to different server instances, replacing or updating hardware for dependent systems such as directory servers or database servers. Try to reuse as many of the configuration settings as possible from the original system on the second system to reduce any configuration issues that might arise in the upgrade.
- On the second system, run all upgrade tasks that might alter data in a production system.

- Typically, the file stores are also relocated to the new platform. If you do not relocate your file stores, you must take extra steps to ensure that the file stores can be accessed from the new system.
- Conduct various validation tests that use the production applications on the upgraded replica system.

Migration template

The migration template lists the major steps that are required to upgrade FileNet P8 onto a different set of servers. Use this template as a starting point for your own plan.

Table 1. Steps required to complete a migration upgrade.

	Migration task.	**Where to go for instructions.**
☐	Learn about upgrading FileNet P8.	https://www.ibm.com/support/knowledgecenter/en/SSNW2F_5.5.0/com.ibm.p8.planprepare.doc/p8ppu006.dita
☐	Create a replica of your FileNet P8 environment. This scenario involves installing a new environment with the same servers and same version of FileNet P8 as your production system. • The GCD, object store, and Process Engine databases in the replica must be backups of the databases that are used on the production system. If the database accounts required by FileNet P8 are not included in the backup, create new instances of the accounts and use the same ID and password as on your production system. • The replica must use the same LDAP provider, LDAP configuration settings, and LDAP-based security accounts as the production system.	• Version 5.5.x: https://www.ibm.com/support/knowledgecenter/SSNW2F_5.5.0/com.ibm.p8toc.doc/welcome_p8.htm • Version 5.5.x: https://www.ibm.com/support/knowledgecenter/SSGLW6_5.5.0 • Version 5.2.1: https://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.1/com.ibm.p8toc.doc/welcome_p8.htm • Version 5.2.1: https://www.ibm.com/support/knowledgecenter/SSGLW6_5.2.1/com.ibm.p8toc.doc/welcome_icf.htm • Version 5.2.0: https://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.0/com.ibm.p8toc.doc/filenetcontentmanager_5.2.0.htm • Version 5.2.0: https://www.ibm.com/support/knowledgecenter/SSGLW6_5.2.0/com.ibm.p8toc.doc/filenetcontentmanager_5.2.0.htm
☐	Prepare the replica for upgrade. Follow the steps in the upgrade preparation instructions. Depending on your replica system, not all upgrade preparation instructions apply to migration upgrades.	Chapter 4, “Preparing the environment to upgrade an existing FileNet P8 domain,” on page 99

Table 1. Steps required to complete a migration upgrade. (continued)

	Migration task.	**Where to go for instructions.**
❐	Upgrade the replica. Follow the steps in “Using containers” to upgrade an existing P8 Domain using containers.	“Upgrading an existing P8 domain to a containers-only environment” on page 120
❐	When the replica system is tested and is ready for production, disconnect the former system and make the replica your new production system.	Putting the upgraded FileNet P8 system into production

Tip: Because large-system upgrades involve interacting with many system components, a common approach is to go through a test upgrade run first. In this scenario, you complete the upgrade on a test system, and then discard the test system after you verify the integrity of the upgrade. You preserve the original production system in case the upgrade fails and to minimize outages to the production system. Running a test upgrade first takes more time than doing the upgrade immediately, but it can minimize risk. The test upgrade approach involves extra steps to completely replicate the production data. Replication is needed to simulate production activities by using a duplicated system.

You can practice the upgrade of your production environment by using either of these two methods:

- Upgrade the lower environments first by running practice upgrades in the following order:
  - Upgrade the development environment
  - Upgrade the various test environments
  - Upgrade the production environment

  This method tests the process and validate that your applications are functioning correctly at each environment level before you move production to the next level.
- Practice the upgrade of the production environment by using the new production environment.
  - Early in the project cycle, copy the existing production databases and file storage areas into the new environment
  - Each time that you practice the upgrade, you apply only the updates that were made to the data since the last practice run.

  This method reduces the time that is needed to complete the final upgrade in the required maintenance window. All of the required FileNet P8 and custom application software is already installed in the new environment. You run only incremental changes to the replicas of databases and file storage areas.

Roadmap for an all-in-one container environment

In the all-in-one container environment, the underlying software and setup are included and completed by the installation tool.

Note: The all-in-one container environment is for a demonstration server or as a non-production developer workstation only. Do not use this scenario for a production environment.

The following roadmap presents a version of the preparation and deployment steps for the all-in-one content services container environment:

1. Verify that you have valid login credentials for the following image sources:
   - The Docker hub
   - The Docker store
   - The IBM Github repository

- IBM Passport Advantage

2. Download all of the images that are required for the environment.
3. Point the installation tool at the directory where you downloaded the required images, and run the installation.

Chapter 3. Preparing the environment for container deployments

Choose the environment and installation scenario that best suits your needs. Then prepare the environment for FileNet P8 Platform container deployment.

About this task

The steps you take to prepare your environment vary depending on the supported platform, tools, and methodology you choose to perform your deployment. In any case, the content services containers require the installation of prerequisite software.

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 IBM Business Automation Configuration Container tool, or into the `jobs.yml` file for deployment and configuration. For lists of the parameters that you need to collect, see the following section: "(V5.5.3 and earlier) Configuration reference" on page 222.

You can choose from the following installation scenarios:

All-in-one demonstration or developer workstation server
: This installation type relies on the all-in-one installation tool that installs all required software and supporting processes automatically. This type is best for demonstration environments or developer workstations and is not useable for a production environment. You do not need to prepare for this type of installation.

(V5.5.4 and later) Deployment with operators
: 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.

IBM Cloud Private
: Using IBM Cloud Private gives extra flexibility in the method for deploying and configuring the containers. You configure and deploy your containers by using the IBM Business Automation Configuration Container configuration tool that is provided with the P8 platform containers, or the command line version of the tool.

Certified Kubernetes server
: Installing on a Kubernetes server environment can be accomplished by using YAML files with the Kubectl command line tool or by using Helm commands. In this scenario, you set up your Kubernetes server and Docker server before you deploy the containers.

For a new P8 domain environment, worker nodes, a directory server, a database installation, and storage are required. Prepare these prerequisites before you begin the configuration and deployment of your containers.

Configuring worker nodes

To deploy any content services component as a container, you must provide a computing environment that meets the minimum requirements.

About this task

A worker node is a node that provides a containerized environment for running tasks. As demands increase, more worker nodes can easily be added to your cluster to improve performance and efficiency. Each worker node that will host an instance or replica of a content services container must meet the hardware and operating system requirements for the product component provided by the individual container.

When you prepare your environment, record the settings so that these values are available to enter into the IBM Business Automation Configuration Container tool, or into the `jobs.yml` file for deployment and configuration. For lists of the parameters that you need to collect, see the following section: "(V5.5.3 and earlier) Configuration reference" on page 222.

Check the IBM Software Product Compatibility Report for the appropriate versions of supporting software.

Procedure

- To prepare your worker node to host a Content Platform Engine or Content Search Services container, follow the procedures in the following topics:
  - Configuring Content Platform Engine servers (AIX®, Linux®, and Linux on System z®)
  - Configuring operating system elements

Preparing the P8 Platform database

You prepare the database to support your P8 Platform components by creating a database user and creating the Global Configuration Database and the object store database. This task applies only when you are preparing to deploy containers as part of a new FileNet P8 domain.

About this task

Check the IBM Software Product Compatibility Report for the appropriate versions of supporting software.

The database that supports your P8 Platform container environment is similar to the traditional on-premises environment. To set up your database, you create a database user, install the database if necessary, and create the Global Configuration Database and the object store database.

When you prepare your environment, record the settings so that these values are available to enter into the IBM Business Automation Configuration Container tool, or into the `jobs.yml` file for deployment and configuration. For lists of the parameters that you need to collect, see the following sections:

- "(V5.5.4 and later) Configuration reference for operators" on page 187
- "(V5.5.3 and earlier) Configuration reference" on page 222

Procedure

To prepare your database:

- Follow the steps for your database type:

Database type	Configuration steps
Db2	Use the following information to prepare your Db2 environment: 1. Create a Content Platform Engine user for Db2 for Linux, UNIX, and Windows. 2. Prepare the Db2 for Linux, UNIX, and Windows server
Oracle	Use the following information to prepare your Oracle environment: 1. Create a Content Platform Engine database user for Oracle. 2. Prepare Oracle server
(V5.5.3 and later) Microsoft SQL Server	Use the following information to prepare your Microsoft SWL Server environment:

Database type	Configuration steps
	1. Creating a Content Platform Engine database user for SQL Server 2. Preparing Microsoft SQL Server

Preparing the P8 Platform directory server

You set up and configure a directory server to provide the authentication repository for your P8 Platform container environment. This task applies only when you are preparing to deploy containers as part of a new FileNet P8 domain.

About this task

Check the IBM Software Product Compatibility Report for the appropriate versions of supporting software.

This procedure assumes that you have installed and prepared a directory service provider that can be used by your container environment.

Note: IBM virtual member manager is not supported for container environments.

When you prepare your environment, record the settings so that these values are available to enter into the IBM Business Automation Configuration Container tool, or into the `jobs.yml` file for deployment and configuration. For lists of the parameters that you need to collect, see the following sections:

- "(V5.5.4 and later) Configuration reference for operators" on page 187
- "(V5.5.3 and earlier) Configuration reference" on page 222

Procedure

To prepare your directory server:

- Follow the steps for your directory server type:

Directory server type	Configuration steps
IBM Security Directory Server	Configure IBM Security Directory Server
Windows Active Directory	Configure Windows Active Directory

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 FileNet P8 domain.

About this task

When you prepare your environment, record the settings so that these values are available to enter into the IBM Business Automation Configuration Container tool, or into the `jobs.yml` file for deployment and configuration. For lists of the parameters that you need to collect, see the following sections:

- "(V5.5.4 and later) Configuration reference for operators" on page 187
- "(V5.5.3 and earlier) Configuration reference" on page 222

Preparing the IBM Content Navigator database on Db2

To prepare your Db2® database for IBM Content Navigator, download and run the database preparation scripts.

About this task

(V5.5.2 and later) The script calls additional scripts to create the database and configure the database. The pre-configuration piece is not required for V5.5.2 (IBM Content Navigator 3.0.5) and later. If you prefer not to have the configuration performed, you can edit the database creation scripts to remove the configuration command. For example, for Db2, you can edit your copy of the `createICNDB.sh` file to remove `DB2_ONE_SCRIPT_ICNDB.sql`.

There is no effect on the deployment if you choose to let the database creation script run the pre-configuration.

Procedure

To prepare the IBM Content Navigator database on Db2:

1. As the Db2 OS user, for example, db2inst1, create a staging folder on the Db2 server.
2. Navigate to https://github.com/ibm-ecm/container-navigator/blob/master/resources/, and download the following scripts to the staging folder that you created:
   - `createICNDB.sh`
   - `DB2_CREATE_SCRIPT.sql`
   - `DB2_ONE_SCRIPT_ICNDB.sql`
3. Grant the proper privileges to the shell script:

   ```
   chmod 755 createICNDB.sh
   ```

4. On the Db2 server, run the following shell script, replacing the parameters with the values for your system:

   ```
   su <db2_os_user>
   ./createICNDB.sh -n <database_name> -s <schema_name>
   -t <tablespace_name> -u <database_user> -a <navigator_admin_id>
   ```

   For example:

   ```
   su db2inst1
   ./createICNDB.sh -n ICNDB -s ICNSCHEMA
   -t ICNTS -u db2inst1 -a p8admin
   ```

   The following table describes the parameters:

Name	Description	Required
n	Navigator database name	Yes
s	Navigator database schema name	Yes
t	Navigator database table space name	Yes
u	Navigator database user name	Yes
a	Navigator admin ID	Yes

Preparing the IBM Content Navigator database on Oracle

To prepare your Oracle database for IBM Content Navigator, download and run the database preparation scripts.

About this task

(V5.5.2 and later) The script calls additional scripts to create the database and configure the database. The pre-configuration piece is not required for V5.5.2 (IBM Content Navigator 3.0.5) and later. If you prefer not to have the configuration performed, you can edit the database creation scripts to remove the configuration command. For example, for Db2, you can edit your copy of the `createICNDB_ORACLE.sh` file to remove `ORACLE_ONE_SCRIPT_ICNDB.sql`.

There is no effect on the deployment if you choose to let the database creation script run the pre-configuration.

Procedure

To prepare the IBM Content Navigator database on Oracle:

1. Create an Oracle database for IBM Content Navigator, for example, icndb.
2. Create the Oracle user, for example, p8user.
3. Navigate to https://github.com/ibm-ecm/container-navigator/blob/master/resources/, and download the following scripts to the staging folder that you created:
   - `createICNDB_ORACLE.sh`
   - `ORACLE_ONE_SCRIPT_ICNDB.sql`
4. Grant the proper privileges to the shell script:

   ```
   chmod 755 createICNDB_ORACLE.sh
   ```

5. On the Oracle server, run the following shell script, replacing the parameters with the values for your system:

   ```
   ./createICNDB_ORACLE.sh -s <database_user> -p <password>
   -r //<oracle_server_ip>:<oracle_server_port>/<database_name> -t <tablespace_name>
   -a <navigator_admin_id>
   ```

   For example:

   ```
   /createICNDB_ORACLE.sh -s p8user -p password -r //localhost:1521/icndb
   -t ICNTS -a p8admin
   ```

   The following table describes the parameters:

Name	Description	Required
s	Oracle user (schema) name	Yes
p	Oracle user password	Yes
r	URL for connecting to Oracle	Yes
t	Navigator database tablespace name	Yes
a	Navigator admin ID	Yes

(V5.5.3 and later) Preparing the IBM Content Navigator database on Microsoft SQL Server

To prepare your SQL Server database for IBM Content Navigator, create the database in SQL Server Management Studio.

Procedure

To prepare the IBM Content Navigator database on SQL Server:

1. In the SQL Server Management Studio, create a Microsoft SQL Server database for IBM Content Navigator, for example, icndb.
2. Create a schema name in SQL Server Management Studio.
3. Expand the database that you created, for example, icndb, and right-click the **Security** folder.
4. Click **New** > **Schema**.
5. Enter the name of the schema that you created and click **OK**.
6. Save the changes for your new database.

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 creating and managing persistent volumes in IBM Cloud Private, see the discussion of storage in the IBM Cloud Private Knowledge Center.
- For more information about persistent volumes, refer to the Kubernetes documentation for persistent volumes.

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.

The PV and PVC are created as a pair with the PVC being used during deployment to request storage for the container. For the content services containers, the persistent volumes 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.
- 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.
- 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.
- 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.

When you prepare your environment, record the settings so that these values are available to enter into the IBM Business Automation Configuration Container tool, or into the `jobs.yml` file for deployment and configuration. For lists of the parameters that you need to collect, see the following sections:

- "(V5.5.4 and later) Configuration reference for operators" on page 187
- "(V5.5.3 and earlier) Configuration reference" on page 222

Preparing storage for FileNet P8 file storage areas and index areas

To store your FileNet P8-managed documents in your file system, you must provide a persistent volume (PV) and persistent volume claim (PVC) for the Content Platform Engine container to use during deployment. To use the full text indexing feature of the Content Search Services, you set up and configure an additional PV and PVC pair to prepare for the Content Search Services container deployment.

About this task

If you want to use Content Search Services, you must provide storage for the Content Search Services container to build the full text indexes. If you choose to store your document content using a file storage area, you can use the following with the Content Platform Engine container to support your FileNet P8 content storage needs:

- File system
- Advanced storage devices:
  - IBM Cloud Object Storage S3
  - Amazon Web Services S3
  - File system

Check the IBM Software Product Compatibility Report for storage hardware and file systems requirements as well as the appropriate versions of supporting software.

Procedure

To prepare storage for storage areas and index areas:

1. To prepare your file system storage to be used as an advanced storage area device, review the information in the following topics:
   - Replication models for advanced storage areas
   - Preparing advanced storage areas
2. To prepare your file system storage to be used as an advanced storage area device, use the following procedures: Preparing file servers for file storage areas

   Note:

   - Follow the suggested best practice of creating a subdirectory under a top-level directory for each advanced storage area device or file storage area you intend to create. If you do not follow this best practice, and you later have to create another top-level directory for an additional device or area, you must rearrange the storage or use other techniques to provide a single top-level directory. A single top-level directory is needed for the persistent volume definition that corresponds to the Content Platform Engine filestore volume for the Content Platform Engine container deployment.
   - Set permissions that allow the Content Platform Engine container to access the file system. The top-level directory and all of its subfolders must be owned by the user, recorded earlier as the value for cpe_os_user, and the group, recorded earlier as cpe_os_group. The cpe_os_user:cpe_os_group numerical IDs vary by container version; set them accordingly:
     - V5.5.2 and later - 50001:50000
     - V5.5.1 - 501:500
3. To prepare your file system storage to be used as an index area, use the following procedures: Preparing for IBM Content Search Services

Note:

- Create a subdirectory under a top-level directory for each object store you intend to enable for full text indexing. You may want to create a hierarchy where there is a top-level directory with subdirectories for each object store, then each object store subdirectory having a subdirectory for each index area that will be created. If you do not follow this guidance, and you later have to create another top-level directory for an index area, you will need to rearrange the storage or use other techniques to provide a single top-level directory. A single top-level directory is needed for the persistent volume definition that corresponds to the index volume for content index data for the Content Search Services container deployment.
- Set permissions that allow the Content Platform Engine container to access the file system. The top-level directory and all of its subfolders must be owned by the user, recorded earlier as the value for cpe_os_user, and the group, recorded earlier as cpe_os_group. The cpe_os_user:cpe_os_group numerical IDs vary by container version; set them accordingly:
  - V5.5.2 and later - 50001:50000
  - V5.5.1 - 501:500

4. Record for later use by the P8 administrator any pre-created subdirectories that will hold advanced storage area devices or index areas data.

   Regardless of the file system that is actually visible from the worker nodes, the content services components running as containers view the file system from an inside-the-container perspective. When using the FileNet P8 administration tools and other configuration interfaces to provide information such as root paths or locations of files, the fully qualified path uses the mountPath value and not the actual path.

   Example 1: Advanced storage area device

   A marketing department has requested a content services deployment. The top-level directory for the file storage areas to use with this deployment is created as an NFS share with a mount path of `/data/FNdomain_Marketing`.

   Two subdirectories are created as `/data/FNdomain_Marketing/ASAdevice1` and `/data/FNdomain_Marketing/ASAdevice2`.

   The PV and PVC pair for the Content Platform Engine filestore volume that is used for the Content Platform Engine container deployment claims `/data/FNdomain_Marketing`.

   The mountPath value for the Content Platform Engine file store volume is `/opt/ibm/asa`.

   When later creating an advanced storage area device object in the object store that needs a file storage area, the root path given by the P8 administrator will be `/opt/ibm/asa/ASAdevice1`.

   Example 2: Index area

   In the same content services deployment for the marketing department, the top-level directory for the index areas was created as an NFS share with a mount point of `/indexes/FNdomain_Marketing`.

   Two subdirectories are created as `/indexes/FNdomain_Marketing/Docs1` and `/indexes/FNdomain_Marketing/Docs2`.

   The PV and PVC pair for the index volume for content index data that is used for the Content Search Services container deployment claims `/indexes/FNdomain_Marketing`.

   The mountPath value for the index volume for content index data is `/opt/ibm/indexareas`.

   When later creating an index area in the object store that is enabled for full text indexing, the root path given by the P8 administrator will be `/CSSIndex1_OS1/Docs1`.

   More information on the inside mountPath values can be found in the topic on Creating volumes and folders for deployment.

Preparing for external key management

You can manage your encryption keys by using an external key management service with a Content Platform Engine container environment.

Before you begin

Prepare the storage environment, including the persistent volumes and persistent volume claims, for your Content Platform Engine container deployment.

About this task

Using external key management for your container deployment of Content Platform Engine requires the following configuration steps:

- Setting up the external key management service, using IBM Key Protect or a supported KMIP-based service.
- Configuring your container environment for external key management.
- Creating your initial domain with the external key management configuration settings.

Use one of the following tasks to prepare for external key management.

Using IBM Key Protect for external key management

You can manage your encryption keys by using IBM Key Protect with a Content Platform Engine container environment.

About this task

The steps to prepare for external key management with IBM Key Protect vary depending on the version of Content Platform Engine that you want to deploy.

(V5.5.4 and later) Preparing for IBM Key Protect for external key management
For V5.5.4 and later, you create a secret with your IBM Key Protect certificate before preparing to deploy your Content Platform Engine container environment.

Procedure

To configure external key management in a container environment that is deployed by an operator:

1. Contact the key management service administrator to acquire the IBM Key Protect certificate or the instructions to download the certificate.
2. Save the certificate file to a location in your container environment.
3. From the directory where the downloaded certificate file is located, run the following command:

   ```
   kubectl create secret generic <secret name> --from-file=tls.crt=<CERT FILE_NAME> -n
   <namespace>
   ```

   Note: The `tls.crt` component is part of the command and must not be changed.
4. After the secret is created, edit the CR YAML file to add the name of the new secret.

   Under the `shared_configuration` section, find the following parameters:

   ```
   root_ca_secret: icp4a-root-ca
   sc_deployment_platform: OCP
   trusted_certificate_list: [<secret name>]
   ```

   Add the secret name as the value for `trusted_certificate_list`, and include the square brackets around the secret name.

What to do next

Continue to prepare for operator deployment by providing the other required values in the CR YAML file, then deploy the Content Platform Engine container image.

When the Content Platform Engine container is up and running, use the Administration Console for Content Platform Engine in the new environment to create a FileNet P8 domain and enable external key management for the domain. For more information, see Creating the FileNet P8 domain.

(V5.5.3) Preparing for IBM Key Protect for external key management

You set up your Content Platform Engine container environment for external key management by using the public key store from the JDK for SSL configuration.

Procedure

To configure external key management in a container environment:

1. Set up the key management service that provides the external key management repository for your FileNet P8 system.

   Use the following instructions to configure the key management service: Using IBM Key Protect for external key management
2. Obtain the `cacerts` file from the JDK, and add the file to the `/configDropins/overrides` directory in the **`cpe-cfgstore`** volume.
3. Create or update the `cpe-ssl.xml` file with the Key Protect configuration:

   - If the `cpe-ssl.xml` file exists in the `/configDropins/overrides` directory for the Content Platform Engine container deployment. add the following configuration details:

     ```
     <server description="custom SSL configuration">
     <ssl id="defaultSSLConfig" trustStoreRef="kpTrustStore"/>
     <keyStore id="kpTrustStore"
     location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/cacerts"
     type="JKS" password="changeit" />
     </server>
     ```

   - If the `cpe-ssl.xml` file does not exist in the `/configDropins/overrides` directory, create the file in that directory with the following content:

     ```
     <?xml version="1.0" encoding="UTF-8"?>
     <server description="custom SSL configuration">
     <ssl id="defaultSSLConfig" trustStoreRef="kpTrustStore"/>
     <keyStore id="kpTrustStore"
     location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/cacerts"
     type="JKS" password="changeit" />
     </server>
     ```
4. Save the `cpe-ssl.xml` file to the configuration overrides directory for the Content Platform Engine container deployment.
5. From the configuration overrides directory, change the permission on the two new files:

   ```
   chown 50001:50000 cacerts
   chown 50001:50000 cpe-ssl.xml
   ```

What to do next

Deploy the Content Platform Engine container image.

When the Content Platform Engine container is up and running, use the Administration Console for Content Platform Engine in the new environment to create a FileNet P8 domain and enable external key management for the domain. For more information, see Creating the FileNet P8 domain.

Using KMIP for external key management

You can manage your encryption keys by using a KMIP external key management service with a Content Platform Engine container environment.

Procedure

To configure external key management in a container environment:

1. Set up the key management service that provides the external key management repository for your FileNet P8 system.

   Use the following instructions to configure the key management service: Using KMIP for external key management
2. Obtain the `cpeTrustStore` and `cpeKeyStore.jceks` files. Add them to the `/configDropins/overrides` directory in the **`cpe-cfgstore`** volume.
3. Create or update the `cpe-ssl.xml` file with the KMIP configuration:
   - If the `cpe-ssl.xml` file exists in the `/configDropins/overrides` directory for the Content Platform Engine container deployment. add the following configuration details:

     ```
     <server description="custom SSL configuration">
     <ssl id ="kmipSSLSettings" keyStoreRef="cpeKeyStore"
     trustStoreRef="cpeTrustStore" clientKeyAlias="client"></ssl>
     <keyStore id="kmipKeyStore"
     location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
     cpeKeyStore.jceks"
     type="JCEKS" password="password" />
     <keyStore id="kmipTrustStore"
     location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/cpeTrustStore"
     type="JCEKS" password="password" />
     </server>
     ```
   - If the `cpe-ssl.xml` file does not exist in the `/configDropins/overrides` directory, create the file in that directory with the following content:

     ```
     <?xml version="1.0" encoding="UTF-8"?>
     <server description="custom SSL configuration">
     <ssl id ="kmipSSLSettings" keyStoreRef="cpeKeyStore"
     trustStoreRef="cpeTrustStore" clientKeyAlias="client"></ssl>
     <keyStore id="kmipKeyStore"
     location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
     cpeKeyStore.jceks"
     type="JCEKS" password="password" />
     <keyStore id="kmipTrustStore"
     location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/cpeTrustStore"
     type="JCEKS" password="password" />
     </server>
     ```

   The **`ssl id`** setting must be `kmipSSLSettings`. The keystore and truststore file names can be named differently, if necessary, but then you must modify the location with the correct name. The client key alias value is used in the keystore.
4. Save the file in the configuration overrides directory for the Content Platform Engine container deployment.
5. From the configuration overrides directory, change the permission on the three new files:

   ```
   chown 50001:50000 cpeTrustStore
   chown 50001:50000 cpeKeyStore.jceks
   chown 50001:50000 cpe-ssl.xml
   ```

What to do next

Deploy the Content Platform Engine container image.

When the Content Platform Engine container is up and running, use the Administration Console for Content Platform Engine in the new environment to create a FileNet P8 domain and enable external key management for the domain. For more information, see Creating the FileNet P8 domain.

Preparing for logging and monitoring

Content services containers provide logging and monitoring capabilities for troubleshooting, performance monitoring, and capacity planning based on monitoring data in real production environments.

About this task

To simplify the enablement of these capabilities, the logging and monitoring components are built into each container product image. You can start using monitoring and logging capabilities very easily by providing a few required environment variables and volumes.

With the logging capability, you can ingest the system log, the Liberty log, and product logs into the back-end logging service that you choose.

With the monitoring capability, you can collect system metrics like CPU, memory, IO, and network data that is related to the content services products through the PCH monitor plug-in, and Liberty runtime information through the JMX plug-in.

You can use Open source Elasticsearch, Logstash, and Kibana (ELK) stack for logging service, and Graphite/Grafana/Carbon for monitoring service

This information helps you to create and collect the settings you need to enable the containers' logging and monitoring capabilities at deployment time.

When you prepare your environment, record the settings so that these values are available to enter into the IBM Business Automation Configuration Container tool, or into the `jobs.yml` file for deployment and configuration. For lists of the parameters that you need to collect, see the following sections:

- “(V5.5.4 and later) Configuration reference for operators” on page 187
- “(V5.5.3 and earlier) Configuration reference” on page 222

Preparing for logging and monitoring for open source

You can use an Elasticsearch/Logstash/Kibana (ELK) stack for logging in your container deployment, and Graphite/Grafana/Carbon for monitoring service.

Before you begin

To set up logging and monitoring for your containers, the following access is required:

- Access to a hosted local ELK stack for logging service and Graphite/Grafana/Carbon stack for the monitoring service

About this task

This information helps you to create and collect the settings you need to enable the containers' logging and monitoring capabilities at deployment time.

Procedure

To configure open source logging and monitoring for the container deployment:

1. Ensure that you have an installation of the ELK stack for the logging service.
2. Ensure that you have an installation of Graphite/Grafana/Carbon for the monitoring service.
3. If your self-hosted Logstash service requires SSL certificate authentication, generate the SSL key and certificate file like by using a command like the following one.

```
openssl req-config /etc/ssl/openssl.cnf -x509 -days 3650
 -batch -nodes -newkey rsa:2048 -keyout private/logstash.key
-out certs/logstash.crt
```

   Important: The certificate name must be `logstash.crt`.

4. Create a persistent volume claim named logmon-cert-cfgstore and copy your generated logstash.crt to that volume.
5. Compile the values for your logging and monitoring system.

   You specify these values in the configuration tool fields for monitoring and logging, or in the YAML file when deploying on the command line.

   Use the following reference information to compile the values:

   - “Monitoring parameters” on page 231
   - “Logging parameters” on page 230

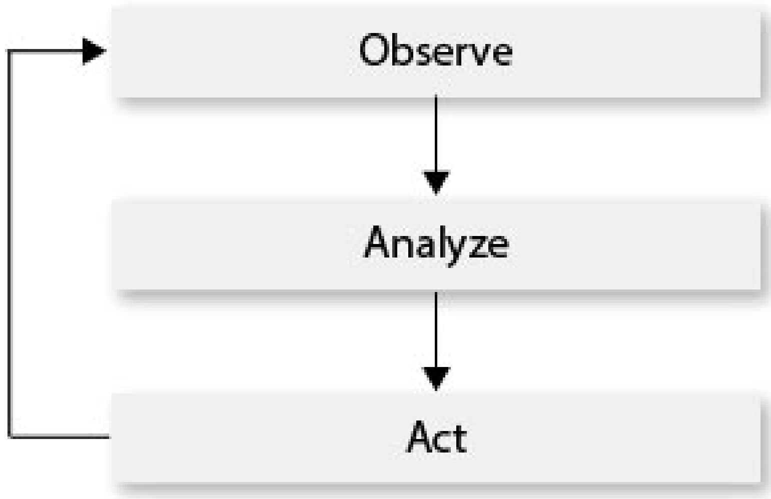
(V5.5.4 and later) Preparing for deployment with an operator

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 FileNet Content Manager 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.

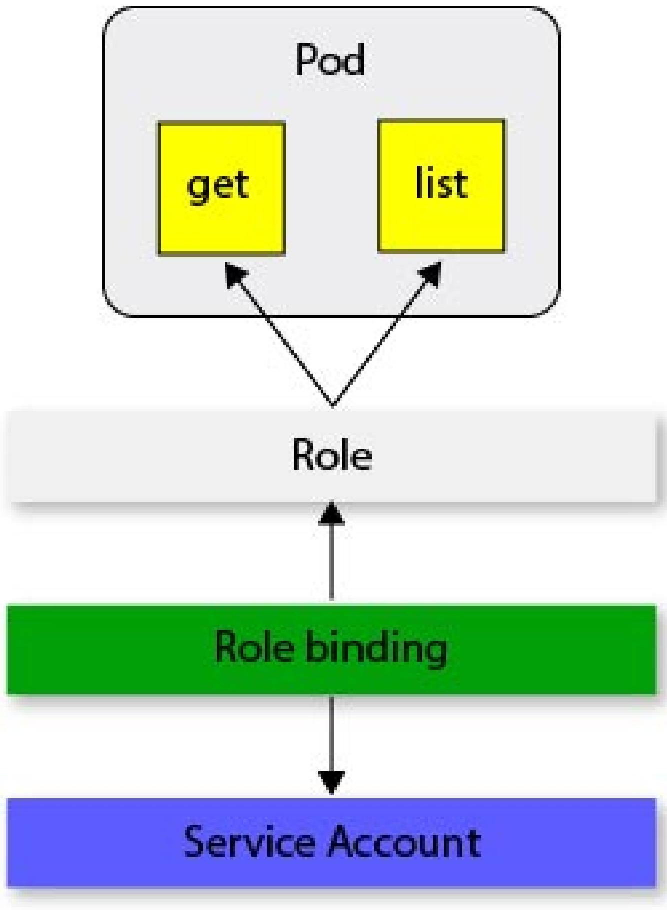
If you deploy all of your Content Services capabilities at the same lifecycle stage into a targeted namespace, you can meter your deployments by the namespace and the platform label or **`swidtag`**. Metering uses a system of metadata annotation and aggregation of VPCs, and is critical to help you understand 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.

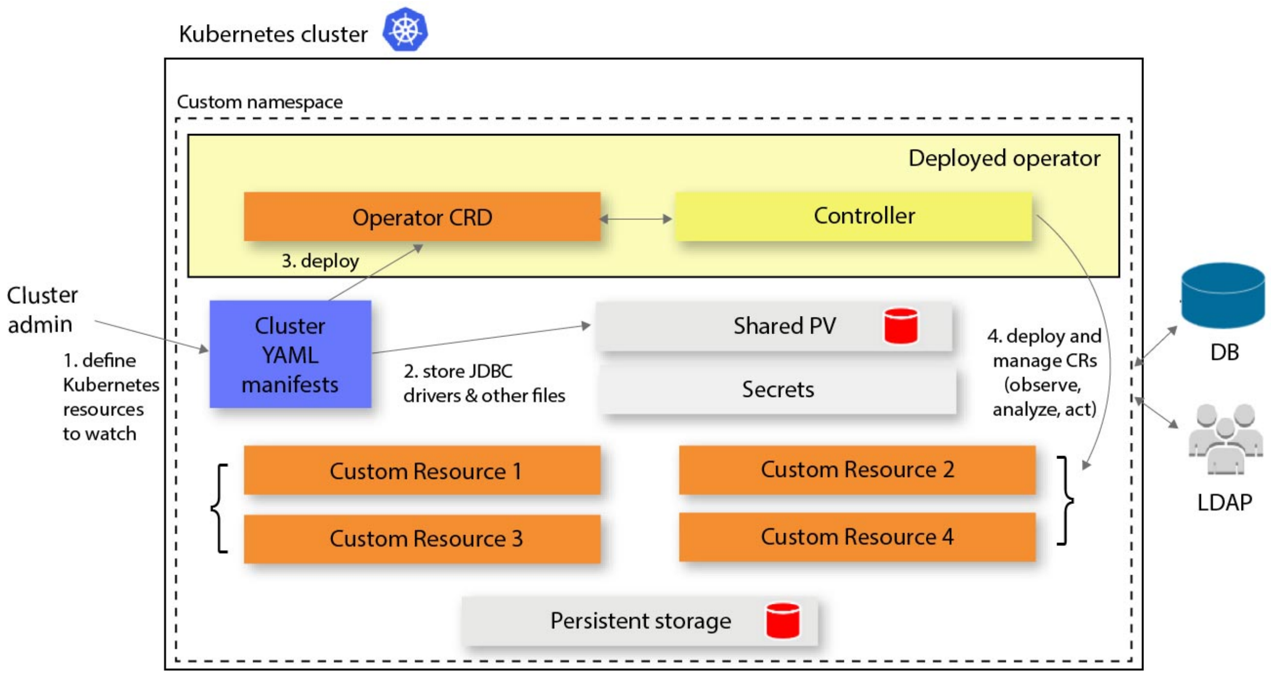
The following diagram shows the relationship between the objects that are used to grant users access to Kubernetes API resources.



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.



Creating volumes and folders for deployment with an operator

The content services component containers require 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

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.

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

Important: Some environments require multiple Content Search Services deployments. To deploy multiple Content Search Services instances, specify a unique release name and service name, and a new set of persistent volumes and persistent volume claims. You can reuse the same persistent volume for the indexstore if you want to have multiple Content Search Services deployments that access the same set of index collections. However, it is recommended that the other persistent volumes be unique.

Procedure

- Create the folders, persistent volumes, and persistent volume claims for the Content Platform Engine container deployment:

  Note: The tables show a single value in the example volume and volume claim name because in most cases the persistent volume and persistent volume claim have the same name.

Table 2. Volumes, volume claims, and folders for Content Platform Engine

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as Seen by Container
Content Platform Engine Liberty configuration	`/configDropins/overrides`	cpe-cfgstore-pv cpe-cfgstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides`
Content Platform Engine Liberty logs	`/logs`	cpe-logstore-pv cpe-logstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/logs`
Content Platform Engine FileNet logs	`/cpefnlogstore`	cpe-fnlogstore-pv cpe-fnlogstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/FileNet`
Content Platform Engine FileNet (upgrade only)	`/bootstrap`	cpe-bootstrapstore-pv cpe-bootstrapstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/lib/Bootstrap`
Content Platform Engine temporary working space	`/textext`	cpe-textextstore-pv cpe-textextstore-pvc	`/opt/ibm/textext`
Content Platform Engine IBM Case Manager rules	`/icmrules`	cpe-icmrulesstore-pv cpe-icmrulesstore-pvc	`/opt/ibm/icmrules`
Content Platform Engine file system-based advanced storage device or file store	`/asa`	cpe-filestore-pv cpe-filestore-pvc	`/opt/ibm/asa`

For each of the folders, set the ownership as follows:

```
chgrp -R 0 /cpecfgstore
```

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

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

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

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-cfgstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/configDropins/overrides
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-cfgstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-cfgstore-pv
  volumeName: cpe-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 Content Platform Engine Liberty logs volume.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-logstore-pv
  volumeName: cpe-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 Content Platform Engine FileNet logs volume.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-filestore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/asa
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-filestore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-filestore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-filestore-pv
  volumeName: cpe-filestore-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 Content Platform Engine bootstrap information (upgrade only).

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-bootstrap-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/bootstrap
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-bootstrap-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-bootstrap-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-bootstrap-pv
  volumeName: cpe-bootstrap-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 Content Platform Engine temporary working space.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-textext-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/textext
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-textext-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-textext-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-textext-pv
  volumeName: cpe-textext-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 Case Manager rules.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-icmrules-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/icmrules
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-icmrules-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-icmrules-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-icmrules-pv
  volumeName: cpe-icmrules-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 Content Platform Engine file system-based advanced storage device or file store.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-filestore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/asa
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-filestore-pv
```

Persistent volume claim:

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

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

Table 3. 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`

Table 3. Volumes, volume claims, and folders for IBM Content Navigator (continued)

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as seen by container
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®	`/icnasperastore`	icn-asperastore-pv icn-asperastore-pvc	/opt/ibm/aspera

For each of the folders, set the ownership as follows:

```
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
```

- Create the persistent volumes and persistent volume claims for the Content Search Services container deployment:

Table 4. Volumes, volume claims, and folders for Content Search Services

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim to Create**	**mountPath as seen by container**
Content Search services container data	`/CSS_Server_data/sslkeystore`	css-cfgstore-pv css-cfgstore-pvc	`/opt/IBM/ContentSearchServices/CSS_Server/data`
Content Search Services temporary working space	`/CSS_Server_temp`	css-tempstore-pv css-tempstore-pvc	`/opt/IBM/ContentSearchServices/CSS_Server/temp`

Table 4. Volumes, volume claims, and folders for Content Search Services (continued)

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim to Create**	**mountPath as seen by container**
Content Search Services logs	`/CSS_Server_log`	css-logstore-pv css-logstore-pvc	`/opt/IBM/ContentSearchServices/CSS_Server/log`
Content Search Services full text indexes	`/CSS_Indexes`	css-indexstore css-indexstore-pvc	`/opt/ibm/indexareas`
Content Search Services custom store	`/csscustomstore/CSS_config`	cs-icp-customstore	`/opt/IBM/ContentSearchServices/CSS_Server/config`

For each of the folders, set the ownership as follows:

```
chgrp -R 0 /csscustomstore
```

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

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

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the Content Search Services container data.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-cfgstore-pv
  volumeName: css-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 Content Search Services temporary working space.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-tempstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-tempstore-pv
  volumeName: css-tempstore-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 Content Search Services logs.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-logstore-pv
  volumeName: css-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 Content Search Services full text indexes.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: css-indexstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/css/CSSIndex_OS1
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: css-indexstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-indexstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-indexstore-pv
  volumeName: css-indexstore-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 Content Search Services custom store.

Persistent volume:

```
apiVersion: v1
{
  "kind": "PersistentVolume",
  "apiVersion": "v1",
  "metadata": {
    "name": "css-icp-customstore-pv",
    "labels": {}
  },
  "spec": {
    "storageClassName": "css-icp-customstore",
    "capacity": {
      "storage": "1Gi"
    },
    "accessModes": [
      "ReadWriteMany"
    ],
    "persistentVolumeReclaimPolicy": "Retain",
    "nfs": {
      "path": "/mnt/nfsshare/css/customstore",
      "server": "<NFS_Server>"
    }
  }
}
```

Persistent volume claim:

```
apiVersion: v1
{
  "kind": "PersistentVolumeClaim",
  "apiVersion": "v1",
  "metadata": {
    "name": "css-icp-customstore",
    "namespace": "default"
  },
  "spec": {
    "storageClassName": "css-icp-customstore",
    "resources": {
      "requests": {
        "storage": "1Gi"
      }
    },
    "accessModes": [
      "ReadWriteMany"
    ]
  }
}
```

- Create the persistent volumes and persistent volume claims for the Content Management Interoperability Services (CMIS) container deployment:

Table 5. Volumes, volume claims, and folders for Content Management Interoperability Services (CMIS)

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim Name**	**mountPath as seen by container**
CMIS and Liberty configuration data	`/configDropins/overrides`	cmis-cfgstore-pv cmis-cfgstore-pvc	`/cmiscfgstore/cmis/configDropins/overrides`
Logs volume for IBM Content Navigator and Liberty logs	`/cmis/logs`	cmis-logstore-pv cmis-logstore-pvc	`/cmislogstore/cmis/logs`

For each of the folders, set the ownership as follows:

```
chgrp -R 0 /cmiscfgstore
```

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

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

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

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cmis-cfgstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cmis/configDropins/overrides
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cmis-cfgstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
```

```
metadata:
  name: cmis-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cmis-cfgstore-pv
  volumeName: cmis-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 CMIS logs.

Persistent volume:

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

Persistent volume claim:

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

- (V5.5.4 and later) Create the persistent volumes and persistent volume claims for the Task Manager container deployment:

Table 6. Volumes, volume claims, and folders for Task Manager

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim to Create**	**mountPath as seen by container**
Task Manager Liberty configuration	`/configDropins/overrides`	tm-cfgstore-pv tm-cfgstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides`

Table 6. Volumes, volume claims, and folders for Task Manager (continued)

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim to Create**	**mountPath as seen by container**
Task Manager and Liberty logs	`/logs`	tm-logstore-pv tm-logstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/logs`
Custom plug-ins for Task Manager	`/plugins`	tm-pluginstore-pv tm-pluginstore-pvc	`/opt/ibm/plugins`

For each of the folders, set the ownership as follows:

```
chgrp -R 0 /tmcfgstore
```

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

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

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the Task Manager configuration store volume.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: tm-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: tm-cfgstore-pv
  volumeName: tm-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 Task Manager and Liberty logs.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
```

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: tm-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: tm-logstore-pv
  volumeName: tm-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 Task Manager plugins.

Persistent volume:

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

Persistent volume claim:

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

Preparing to install with an operator on Kubernetes

Prepare your environment and install the necessary software before you go to the GitHub repositories to find further information on installing the containers with the operator.

Before you begin

Install the following list of software before you install any of the content services 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 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 databases for your intended 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 eAssembly images for Kubernetes.

Preparing to install with an operator on Managed Red Hat OpenShift on IBM Cloud Public

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.

Before you begin

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

1. Create an account on IBM Cloud.

   Log in to IBM Cloud if you already have an account.
2. 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**.

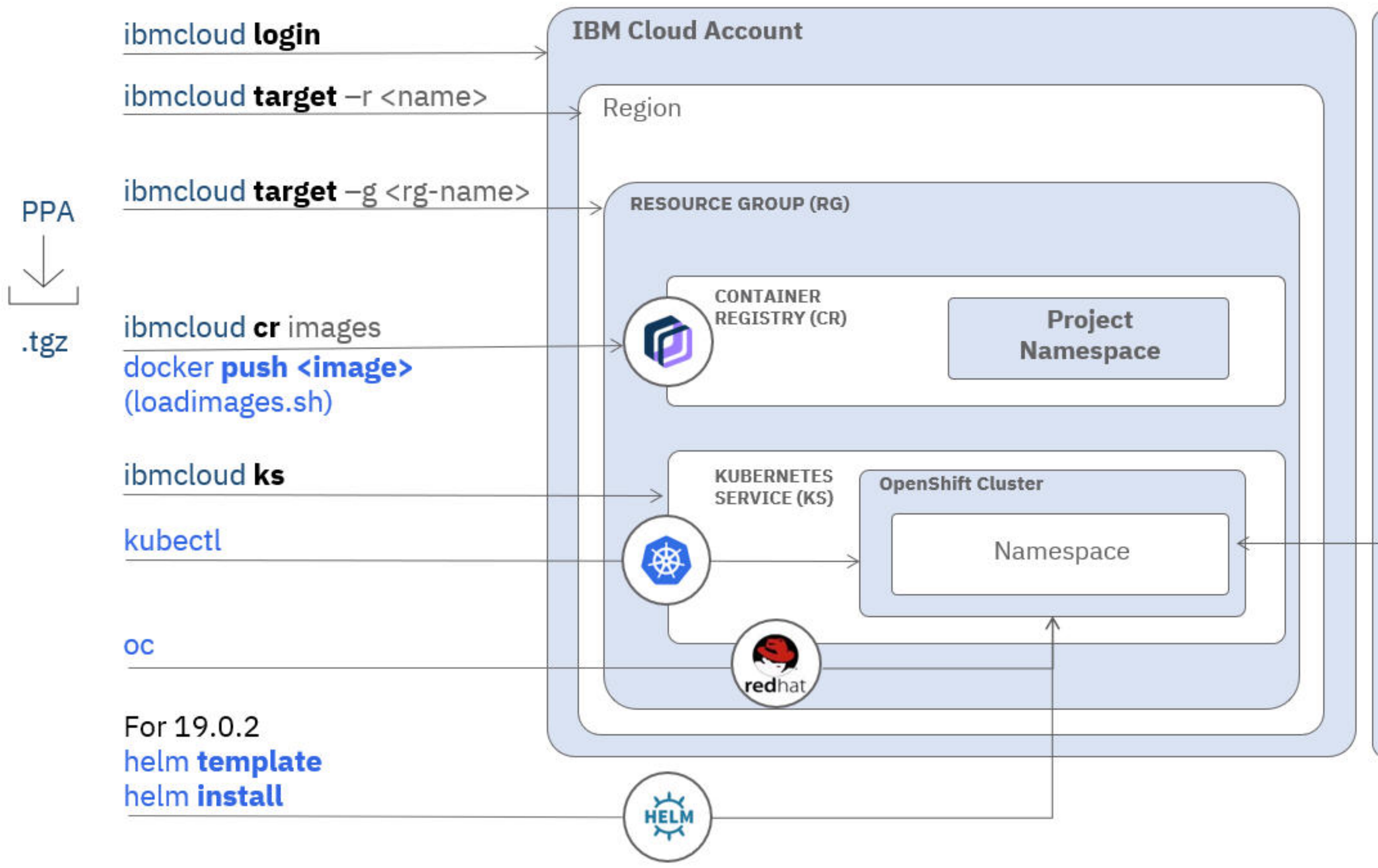
   Refer to the IBM Cloud documentation to create a Kubernetes cluster.

   The cluster that you create includes attached storage.

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 (Windows)

The command line tools provide granular control of the various components of IBM Cloud.



To install IBM certified software for production purposes (production charts) on IBM Cloud Public, you must prepare your environment in 1 of 2 ways before you run the installation commands.

Note: Evaluation charts are accessible on public docker registries so you do not need to create an image pull secret to use these images.

- `Option 1`: Create a secret to the IBM Cloud Entitled Registry. The IBM Cloud Entitled Registry is the container library that is associated with your IBM account (MyIBM). You can pull the container images in your library directly from the command line.
- `Option 2`: Download the software packages from Passport Advantage (PPA), extract the images, upload the images to the IBM Cloud Container Registry, and create a secret to be able to pull the images.

Procedure

1. Log in to your IBM Cloud account and select **Kubernetes** from the menu icon.
2. Select the cluster that you created and from the **cluster details** page, click **OpenShift web console**.
3. In the **OpenShift web console** menu bar, click your profile **IAM#user.name@email.com** > **Copy Login Command** and paste the copied command into your command line.

```
oc login https://<CLUSTERNAME>:<CLUSTERPORT> --token=<GENERATED_TOKEN>
```

4. Create a project for each release you want to install by running the following commands.

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

5. Add privileges to the projects. Grant `ibm-anyuid-scc` privileges to any authenticated user and grant `ibm-privileged-scc` privileges to any authenticated user.

```
oc project <project_name>
oc adm policy add-scc-to-user privileged -z default
oc adm policy add-scc-to-group ibm-anyuid-scc system:authenticated
oc adm policy add-scc-to-user ibm-privileged-scc system:authenticated
```

Note: You need a privileged account to run the `oc adm policy` command. The *<project_name>* must have pull request privileges to the registry where the images are loaded. The *<project_name>* must also have pull request privileges to push the images into another namespace.

6. 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 Certified Kubernetes eAssembly images that you want to install, go to Passport Advantage and find the part numbers in the download document. Download all the **Certified Kubernetes Multiplatform** parts that you want to install.

     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.

(V5.5.3 and earlier) Preparing for deployment on IBM Cloud Private

Prepare your IBM Cloud Private environment for your container deployment.

About this task

Check the IBM Software Product Compatibility Report for the appropriate versions of supporting software.

When you prepare your environment, record the settings so that these values are available to enter into the IBM Business Automation Configuration Container tool, or into the jobs.yml file for deployment and configuration. For lists of the parameters that you need to collect, see the following section: "(V5.5.3 and earlier) Configuration reference" on page 222.

Procedure

To prepare for deployment:

1. Ensure that you have a supported version of IBM Cloud Private installed and configured on your target server.

   For details on setting up IBM Cloud Private, see the following information: Installing IBM Cloud Private

   Consult with your installation administrator on the following requirements:

   - Set up Docker and configure storage before you run the IBM Cloud Private installation.
   - Review the Configuring options during installation section to determine which options are appropriate for your container environment.
   - To use the command line interface, you must install and configure the kubectl tool. Use the following information to complete this installation: Accessing your IBM® Cloud Private cluster by using the kubectl CLI

- To import Helm charts, you must have the Helm CLI tool installed. Use the following information to complete the installation: Setting up the Helm CLI

2. Create a namespace in IBM Cloud Private for your content services deployment.

   Namespaces are required to organize users and their applications. Use the following information to create the namespace: Creating a namespace.

3. Create an imagePullSecrets authorization token in the namespace.

   Use the following information to create the imagePullSecrets: Creating imagePullSecrets for a specific namespace

Creating volumes and folders for deployment on IBM Cloud Private

The content services component containers require a certain number of 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

Although the following 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.

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

Persistent volume:

```
{
  "kind": "PersistentVolume",
  "apiVersion": "v1",
  "metadata": {
    "name": "cpe-icp-cfgstore",
    "labels": {}
  },
  "spec": {
    "storageClassName": "cpe-icp-cfgstore",
    "capacity": {
      "storage": "1Gi"
    },
    "accessModes": [
      "ReadWriteMany"
    ],
    "persistentVolumeReclaimPolicy": "Retain",
    "nfs": {
      "path": "/cpe/cfg",
      "server": "127.0.0.1"
    }
  }
}
```

Persistent volume claim:

```
{
  "kind": "PersistentVolumeClaim",
  "apiVersion": "v1",
  "metadata": {
    "name": "cpe-icp-cfgstore",
    "namespace": "ecm-icp-demo"
  },
  "spec": {
    "storageClassName": "cpe-icp-cfgstore",
    "resources": {
      "requests": {
        "storage": "1Gi"
      }
    },
    "accessModes": [
      "ReadWriteMany"
    ]
  }
}
```

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

Important: Some environments require multiple Content Search Services deployments. To deploy multiple Content Search Services instances, specify a unique release name and service name, and a new set of persistent volumes and persistent volume claims. You can reuse the same persistent volume for the indexstore if you want to have multiple Content Search Services deployments that access the same set of index collections. However, it is recommended that the other persistent volumes be unique.

Procedure

- Create the folders, persistent volumes, and persistent volume claims for the IBM Business Automation Configuration Container deployment:

  Note: The tables show a single value in the example volume and volume claim name because in most cases the persistent volume and persistent volume claim have the same name.

  Table 7. Volumes, volume claims, and folders for the IBM Business Automation Configuration Container

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as Seen by Container
Configuration tool data	`/nfsstorage`	ecm-icp-ibacc	

  (V5.5.2 and later) For each of the folders, set the ownership to 50001:50000:

  ```
  chown –Rf 50001:50000 /cpecfgstore
  ```

  (V5.5.1) For each of the folders, set the ownership to 501:500:

  ```
  chown –Rf 501:500 /ibacc
  ```

- Create the folders, persistent volumes, and persistent volume claims for the Content Platform Engine container deployment:

  Note: The tables show a single value in the example volume and volume claim name because in most cases the persistent volume and persistent volume claim have the same name.

  Table 8. Volumes, volume claims, and folders for Content Platform Engine

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as Seen by Container
Content Platform Engine Liberty configuration	`/cpecfgstore/cpe/configDropins/overrides`	cpe-icp-cfgstore	`/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides`
Content Platform Engine Liberty logs	`/cpelogstore/cpe/logs`	cpe-icp-logstore	`/opt/ibm/wlp/usr/servers/defaultServer/logs`

Table 8. Volumes, volume claims, and folders for Content Platform Engine (continued)

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as Seen by Container
Content Platform Engine FileNet logs	`/cpefnlogstore`	cpe-icp-fnlogstore	`/opt/ibm/wlp/usr/servers/defaultServer/FileNet`
Content Platform Engine FileNet (upgrade only)	`/cpebootstrapstore/bootstrap`	cpe-icp-bootstrapstore	`/opt/ibm/wlp/usr/servers/defaultServer/lib/Bootstrap`
Content Platform Engine temporary working space	`/cpetextextstore/textext`	cpe-icp-textextstore	`/opt/ibm/textext`
Content Platform Engine IBM Case Manager rules	`/cpeicmrulesstore/icmrules`	cpe-icmrulestore	`/opt/ibm/icmrules`
Content Platform Engine file system-based advanced storage device or file store	`/cpefilestore/asa`	cpe-icp-filestore	`/opt/ibm/asa`

(V5.5.2 and later) For each of the folders, set the ownership to 50001:50000:

```
chown –Rf 50001:50000 /cpecfgstore
```

(V5.5.1) For each of the folders, set the ownership to 501:500:

```
chown –Rf 501:500 /ibacc
```

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

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-icp-cfgstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/configDropins/overrides
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-icp-cfgstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-icp-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-icp-cfgstore-pv
  volumeName: cpe-icp-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 Content Platform Engine Liberty logs volume.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-icp-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-icp-logstore-pv
  volumeName: cpe-icp-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 Content Platform Engine FileNet logs volume.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-icp-filestore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/asa
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-icp-filestore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-icp-filestore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-icp-filestore-pv
  volumeName: cpe-icp-filestore-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 Content Platform Engine bootstrap information (upgrade only).

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-icp-bootstrap-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/bootstrap
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-icp-bootstrap-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-icp-bootstrap-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-icp-bootstrap-pv
  volumeName: cpe-icp-bootstrap-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 Content Platform Engine temporary working space.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-icp-textext-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/textext
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-icp-textext-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-icp-textext-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-icp-textext-pv
  volumeName: cpe-icp-textext-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 Case Manager rules.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-icp-icmrules-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/icmrules
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-icp-icmrules-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-icp-icmrules-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-icp-icmrules-pv
  volumeName: cpe-icp-icmrules-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 Content Platform Engine file system-based advanced storage device or file store.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-icp-filestore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/asa
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-icp-filestore-pv
```

Persistent volume claim:

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

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

Table 9. 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	`/icncfgstore/icn/configDropins/overrides`	icn-icp-cfgstore	`/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides`
IBM Content Navigator and Liberty logs	`/icnlogstore/icn/logs`	icn-icp-logstore	`/opt/ibm/wlp/usr/servers/defaultServer/logs`
Custom plug-ins for IBM Content Navigator	`/icnpluginstore/icn/plugins`	icn-icp-pluginstore	`/opt/ibm/plugins`
IBM Content Navigator viewer logs for Daeja ViewONE	`/icnvwlogstore`	icn-icp-vwlogstore	`/opt/ibm/viewerconfig/logs`
IBM Content Navigator storage for the Daeja ViewONE cache	`/icnvwcachestore`	icn-icp-vwcachestore	`/opt/ibm/viewerconfig/cache`
(V5.5.2 and later) IBM Content Navigator storage for Aspera	/icnasperastore	icn-icp-asperastore	`/opt/ibm/aspera`

(V5.5.2 and later) For each of the folders, set the ownership to 50001:50000:

```
chown –Rf 50001:50000 /cpecfgstore
```

(V5.5.1) For each of the folders, set the ownership to 501:500:

```
chown –Rf 501:500 /ibacc
```

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-icp-cfgstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/configDropin/overrides
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-icp-cfgstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-icp-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-icp-cfgstore-pv
  volumeName: icn-icp-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-icp-logstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/logs
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-icp-logstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-icp-logstore-pvc
  namespace: <NAMESPACE>
```

```
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-icp-logstore-pv
  volumeName: icn-icp-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: cn-icp-pluginstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/plugins
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cn-icp-pluginstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-icp-pluginstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-icp-pluginstore-pv
  volumeName: icn-icp-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-icp-vw-logstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/viewerlog
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-icp-vw-logstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-icp-vw-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-icp-vw-logstore-pv
  volumeName: icn-icp-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-icp-vw-cachestore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/viewercache
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-icp-vw-cachestore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: icn-icp-vw-cachestore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: icn-icp-vw-cachestore-pv
  volumeName: icn-icp-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-icp-asperastore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/aspera
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-icp-asperastore-pv
```

Persistent volume claim:

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

- Create the persistent volumes and persistent volume claims for the Content Search Services container deployment:

Table 10. Volumes, volume claims, and folders for Content Search Services

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as seen by container
Content Search services container data	`/csscfgstore/css/CSS_Server_data/sslkeystore`	css-icp-data	`/opt/IBM/ContentSearchServices/CSS_Server/data`
Content Search Services temporary working space	`/csstempstore/CSS_Server_temp`	css-icp-temp	`/opt/IBM/ContentSearchServices/CSS_Server/temp`
Content Search Services logs	`/csslogstore/CSS_Server_log`	css-icp-logstore	`/opt/IBM/ContentSearchServices/CSS_Server/log`
Content Search Services full text indexes	`/cssindexstore/CSS_Indexes`	css-icp-indexes	`/opt/ibm/indexareas`
(V5.5.3 and later) Content Search Services custom store	`/csscustomstore/CSS_config`	cs-icp-customstore	`/opt/IBM/ContentSearchServices/CSS_Server/config`

(V5.5.2 and later) For each of the folders, set the ownership to 50001:50000:

```
chown –Rf 50001:50000 /csscustomstore
```

(V5.5.1) For each of the folders, set the ownership to 501:500:

```
chown –Rf 501:500 /ibacc
```

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the Content Search Services container data.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-icp-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-icp-cfgstore-pv
  volumeName: css-icp-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 Content Search Services temporary working space.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-icp-tempstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-icp-tempstore-pv
  volumeName: css-icp-tempstore-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 Content Search Services logs.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-icp-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-icp-logstore-pv
  volumeName: css-icp-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 Content Search Services full text indexes.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: css-icp-indexstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/css/CSSIndex_OS1
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: css-icp-indexstore-pv
```

Persistent volume claim:

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

(V5.5.3 and later) The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the Content Search Services custom store.

Persistent volume:

```
apiVersion: v1
{
  "kind": "PersistentVolume",
  "apiVersion": "v1",
  "metadata": {
    "name": "css-icp-customstore-pv",
    "labels": {}
  },
  "spec": {
    "storageClassName": "css-icp-customstore",
    "capacity": {
      "storage": "1Gi"
    },
    "accessModes": [
      "ReadWriteMany"
    ],
    "persistentVolumeReclaimPolicy": "Retain",
    "nfs": {
      "path": "/mnt/nfsshare/css/customstore",
      "server": "9.30.166.71"
    }
  }
}
```

Persistent volume claim:

```
apiVersion: v1
{
  "kind": "PersistentVolumeClaim",
  "apiVersion": "v1",
  "metadata": {
    "name": "css-icp-customstore",
    "namespace": "default"
  },
  "spec": {
    "storageClassName": "css-icp-customstore",
    "resources": {
      "requests": {
        "storage": "1Gi"
      }
    },
    "accessModes": [
      "ReadWriteMany"
    ]
  }
}
```

- Create the persistent volumes and persistent volume claims for the Content Management Interoperability Services (CMIS) container deployment:

Table 11. Volumes , volume claims, and folders for Content Management Interoperability Services (CMIS)

Volume purpose	Example Folder to Create	Example Volume and Volume Claim Name	mountPath as seen by container
CMIS and Liberty configuration data	`/cmiscfgstore/cmis/configDropins/overrides`	cmis-icp-cfgstore	`/cmiscfgstore/cmis/configDropins/overrides`
Logs volume for IBM Content Navigator and Liberty logs	`/cmislogstore/cmis/logs`	cmis-icp-logstore	`/cmislogstore/cmis/logs`

(V5.5.3) For each of the folders, set the ownership to 50001:50000:

```
chown –Rf 50001:50000 /cpecfgstore
```

(V5.5.1 and V5.5.2) For each of the folders, set the ownership to 501:500:

```
chown –Rf 501:500 /cmiscfgstore
```

(V5.5.2) Change the mode of the persistent volume to 777:

```
chmod -R 777 /cmis-pv
```

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

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cmis-icp-cfgstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cmis/configDropins/overrides
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cmis-icp-cfgstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cmis-icp-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cmis-icp-cfgstore-pv
  volumeName: cmis-icp-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 CMIS logs.

Persistent volume:

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

Persistent volume claim:

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

Creating and loading the content services archive file

To make the images for content services containers and the IBM Business Automation Configuration Container available for deployment in IBM Cloud Private, you must first create an archive file of the images. You then load the created archive file into IBM Cloud Private.

Before you begin

When running in some PAAS environments, you might need to configure the NO_PROXY setting in advance. Otherwise, you might encounter an error while loading the chart archive.

Run the following command:

```
# export NO_PROXY=localhost,127.0.0.1,domain_suffix,ICP_CLUSTER_NAME,
SC_K8S_CLUSTER_NAME
```

For example:

```
# export NO_PROXY=localhost,127.0.0.1,.suffix.ibm.com,mycluster,mycluster.icp
```

About this task

Important: If you have already created the archive file in order to deploy the IBM Business Automation Configuration Container, you do not need to create another archive file. Your content services container images are already available in IBM Cloud Private because of the upload and deploy steps you took for the IBM Business Automation Configuration Container.

For V5.5.3, see the V5.5.3 download doc to find the part numbers for the components that you need for this task, as well as the link to Passport Advantage: http://www.ibm.com/support/docview.wss?uid=ibm10881658

For V5.5.2, see the V5.5.2 download doc to find the part numbers for the components that you need for this task, as well as the link to Passport Advantage: http://www.ibm.com/support/docview.wss?uid=ibm10741447

For V5.5.1, see the V5.5.1 download doc to find the part numbers for the components that you need for this task, as well as the link to Passport Advantage: https://www.ibm.com/support/docview.wss?uid=swg24044874

Procedure

To create the container image archive file:

1. From Passport Advantage, download the IBM Business Automation Configuration Container compressed file, *ibacc-part-number*.tar.gz, and use the tar command to extract the file to a folder that you specify:

   ```
   tar -zxvf ibacc-part-number.tar.gz -C <output_folder>
   ```

   (V5.5.3) The following example shows the output folder contents:

   ```
   <output_folder>
   ├── charts
   │    └── ibm-dba-content-prod-3.0.0.tgz
   ├── images
   │    ├── a3ea3c3ab12581c6fa853775a4f466db7a91704fa0f7bf5daa04f5badd8c668e.tar.gz
   │    ├── 56587cfdf24ae5bba72efc73d24bd3788ca20a676c997798447c7c03fce330f1.tar.gz
   │    ├── fdba4cd1b6287593f7d3e186d2c013f92d6a7d927b6eb134407b0bc0ec51c2fe.tar.gz
   │    └── 2e88e40f634f4bf82ceb98056759a45356a82116bc695ec2f2ee47d24d0a0119.tar.gz
   ├── manifest.json
   └── manifest.yaml
   ```

   (V5.5.2) The following example shows the output folder contents:

   ```
   <output_folder>
   ├── charts
   │    └── ibm-dba-content-prod-2.0.0.tgz
   ├── images
   │    ├── 6bc9b8634a9af27b45d3315d8a4373979aaa492244c6590c2754df1b40df3858.tar.gz
   │    ├── 78395a35e0e26ba1868c925fcf31ca7965f52a2008d8e1447479a4ef18075938.tar.gz
   │    ├── c5d8397915ffc3a9993b8e621f99d704ae944d0fdf5a9f347e368b3c06c7e8a5.tar.gz
   │    └── f86d4686c177a6d817b3edc0690bb3f9bfd2c9cbe48909172a937a243f7ea802.tar.gz
   ├── manifest.json
   └── manifest.yaml
   ```

   (V5.5.1) The following example shows the output folder contents:

   ```
   <output_folder>
   ├── charts
   │    └── ibm-dba-content-prod-1.2.0.tgz
   ├── images
   │    ├── 6bc9b8634a9af27b45d3315d8a4373979aaa492244c6590c2754df1b40df3858.tar.gz
   │    ├── 78395a35e0e26ba1868c925fcf31ca7965f52a2008d8e1447479a4ef18075938.tar.gz
   │    ├── c5d8397915ffc3a9993b8e621f99d704ae944d0fdf5a9f347e368b3c06c7e8a5.tar.gz
   │    └── f86d4686c177a6d817b3edc0690bb3f9bfd2c9cbe48909172a937a243f7ea802.tar.gz
   ├── manifest.json
   └── manifest.yaml
   ```

   The long file names are the SHA256 IDs of the specified Docker images.

2. From Passport Advantage, download the Docker images for the content services component containers:

   - `*CPE-container-part-number*.tar`
   - `*ICN-container-part-number*.tar`
   - `*CSS-container-part-number*.tar`
   - The CMIS image:
     - (V5.5.3) *Fix-ID*.tar
     - (V5.5.2) *Fix-ID*.tar

  - (V5.5.1) *CMIS-container-part-number*.tar

3. Run the tar command to extract the original content services image archive tar.gz files:

   ```
   # tar xvf CPE-container-part-number.tar
   ```

   Repeat the command for each of the remaining three downloaded content service container image tar files:

   - (V5.5.3) *Fix-ID*.tar
   - *ICN-container-part-number*.tar
   - *CSS-container-part-number*.tar
   - The CMIS tar file:
     - (V5.5.3) *Fix-ID*.tar
     - (V5.5.2) *Fix-ID*.tar
     - (V5.5.1) *CMIS-container-part-number*.tar

   CMIS-container-part-number.tar

   The command results in a tar.gz file from each container TAR file, and two tar.gz files from the IBM Content Navigator container TAR file. Note that the tar.gz files can have long IDs.

4. Move the content services image archive files (*.tar.gz) to the `images` folder in the output folder that you created when you extracted the IBM Business Automation Configuration Container tar file:

   ```
   mv <cpe_image_tar>.gz  <output_folder>/images
   ```

   Repeat the command for each of the remaining archive image files that you extracted. Remember that the IBM Content Navigator TAR file extraction results in two *image*.tar.gz files that must both be moved to the *output_folder*/images folder.

   (V5.5.3) The contents of the output folder now looks like the following example:

   ```
   <output_folder>
       ├── charts
       │    └── ibm-dba-content-prod-3.0.0.tgz
       ├── images
       │    ├── a3ea3c3ab12581c6fa853775a4f466db7a91704fa0f7bf5daa04f5badd8c668e.tar.gz
       │    ├── 56587cfdf24ae5bba72efc73d24bd3788ca20a676c997798447c7c03fce330f1.tar.gz
       │    ├── fdba4cd1b6287593f7d3e186d2c013f92d6a7d927b6eb134407b0bc0ec51c2fe.tar.gz
       │    ├── 2e88e40f634f4bf82ceb98056759a45356a82116bc695ec2f2ee47d24d0a0119.tar.gz
       │    ├── 1ba75e4174939b96b17335a3dab336a96e6fcc3df75d8ac0fcca6d7fdd4c0eb1.tar.gz
       │    ├── 395b70bc1a9944a4d01ac61eb9e78db1be76fad1e067f2110b883a2f684dc4fb.tar.gz
       │    ├── 194262f8351e6349f8d0058cdd00fa8f311fc0962a7ae7e193be248d0b606cea.tar.gz
       │    ├── 0e33b27dd1138f23ce825019e02fe3810091685b8e9de5dce04ae8ee17ed27ae.tar.gz
       │    └── 004bdd4f629a271ec551aeede1c43295ec1cfd766de57fd6d7083c5e35863556.tar.gz
       ├── manifest.json
       └── manifest.yaml
   ```

   (V5.5.2) The contents of the output folder now looks like the following example:

   ```
   <output_folder>
       ├── charts
       │    └── ibm-dba-content-prod-2.0.0.tgz
       ├── images
       │    ├── 5529941196f0ec9327ff5056ab723643a3c094895320b2179a47876e51a15302.tar.gz
       │    ├── 5d59dfb451f7c347660d792c97625b108fe243bccee0498e5bfa862d4e01e4c3.tar.gz
       │    ├── 6bc9b8634a9af27b45d3315d8a4373979aaa492244c6590c2754df1b40df3858.tar.gz
       │    ├── 78395a35e0e26ba1868c925fcf31ca7965f52a2008d8e1447479a4ef18075938.tar.gz
       │    ├── 8fefb76b3abd78f383c35eb49630675925ccfcf6f17d50bf112528b4f57e445c.tar.gz
       │    ├── b343d3e7d3ad9d8d19116376997eefe2569db2f2aa958e279aa3844ea8f9c602.tar.gz
       │    ├── c5d8397915ffc3a9993b8e621f99d704ae944d0fdf5a9f347e368b3c06c7e8a5.tar.gz
       │    ├── d2cba67ac4c1f5f5b9ea14eb2ab010f8fadc625b828e6b8f1d2e1f00e6e0c3f1.tar.gz
       │    └── f86d4686c177a6d817b3edc0690bb3f9bfd2c9cbe48909172a937a243f7ea802.tar.gz
       ├── manifest.json
       └── manifest.yaml
   ```

   (V5.5.1) The contents of the output folder now looks like the following example:

```
<output_folder>
    ├── charts
    │   └── ibm-dba-content-prod-1.3.0.tgz
    ├── images
    │   ├── 5529941196f0ec9327ff5056ab723643a3c094895320b2179a47876e51a15302.tar.gz
    │   ├── 5d59dfb451f7c347660d792c97625b108fe243bccee0498e5bfa862d4e01e4c3.tar.gz
    │   ├── 6bc9b8634a9af27b45d3315d8a4373979aaa492244c6590c2754df1b40df3858.tar.gz
    │   ├── 78395a35e0e26ba1868c925fcf31ca7965f52a2008d8e1447479a4ef18075938.tar.gz
    │   ├── 8fefb76b3abd78f383c35eb49630675925ccfcf6f17d50bf112528b4f57e445c.tar.gz
    │   ├── b343d3e7d3ad9d8d19116376997eefe2569db2f2aa958e279aa3844ea8f9c602.tar.gz
    │   ├── c5d8397915ffc3a9993b8e621f99d704ae944d0fdf5a9f347e368b3c06c7e8a5.tar.gz
    │   ├── d2cba67ac4c1f5f5b9ea14eb2ab010f8fadc625b828e6b8f1d2e1f00e6e0c3f1.tar.gz
    │   └── f86d4686c177a6d817b3edc0690bb3f9bfd2c9cbe48909172a937a243f7ea802.tar.gz
    ├── manifest.json
    └── manifest.yaml
```

5. Change to the *output_folder* location:

```
# cd output_folder
```

6. Run the tar command to compress the `output_folder`:

```
# tar -zcvf icp_charts_images.tgz output_folder/*
```

Important: Ensure that the content and folder and file structure inside the archive file are the same as what is in the *output_folder* location.

7. Use the command line to log in to IBM Cloud Platform:

For V5.5.3, use the following command:

```
cloudctl login -a https://<icp_ip_address>:8443 --skip-ssl-validation
```

For V5.5.2, use the following command:

```
cloudctl login -a https://<icp_ip_address>:8443 --skip-ssl-validation
```

For V5.5.1, use the following command:

```
bx pr login -a https://<icp_ip_address>:8443 --skip-ssl-validation
```

8. Log in to your cluster:

```
docker login mycluster.icp:8500
```

9. Load the archive file that you created into IBM Cloud Private:

For V5.5.3, use the following command:

```
cloudctl catalog load-archive --archive icp_charts_images.tgz --registry mycluster.icp:8500
```

For V5.5.2, use the following command:

```
cloudctl catalog load-archive --archive icp_charts_images.tgz
--clustername mycluster.icp --namespace ecm-namespace

or

cloudctl catalog load-archive --archive icp_charts_images.tgz
--registry mycluster.icp:8500 --namespace your_namespace
```

For V5.5.1, use the following command:

```
bx pr load-ppa-archive --archive icp_charts_images.tgz --clustername your_cluster
--namespace your_namespace
```

In the command, replace any variables for cluster, namespace, and so on with actual values from your environment.

What to do next
(V5.5.3) Verify that the container images that you loaded have the following tags:

- cpe:ga-553-p8cpe
- cpe-sso:ga-553-p8cpe
- css:ga-553-p8css
- navigator:ga-306-icn
- navigator-sso:ga-306-icn
- cmis:ga-304-cmis-if007

Deploying the IBM Business Automation Configuration Container

You use the configuration tool to deploy your content services containers. Deploy the IBM Business Automation Configuration Container in IBM Cloud Private to enable access to the configuration tool.

Before you begin

Before you can deploy the IBM Business Automation Configuration Container, you must create and upload the archive file of container images.

Procedure

To deploy the IBM Business Automation Configuration Container:

1. Open the IBM Cloud Private Admin console and enter the administrator user name and password (admin/admin by default).

   ```
   https://<IP_address>:8443
   ```

   Where *<IP_address>* is the URL in the cfc-master: `POST DEPLOY MESSAGE` String. By default, the host name is `mycluster.icp`.

   The **Welcome** page opens and provides menus to manage your private cloud environment.
2. Click **Catalog** on the menu bar, select the **ibm-dba-content-prod** entry, and click **Configure**.

   The helm chart displays a page of documentation, which describes the purpose of the chart and lists the configuration parameters that can be used to modify the default settings.
3. Provide the namespace that you created for the deployment.
4. Enter a release name and accept the license agreement.
   An example release name is *ibacc-master*.
5. Edit the **Repository for Docker Container Image** field with the namespace and provide the values or use the default values in other fields.

   Verify that the value for the Persistent Volume Claim for the Business Automation Configuration Container is correct.
6. Click **Install**.
7. When the deployment completes, navigate to **Workloads** > **Helm Release** > ***ibacc-master*** > **Services**, and click the name of your IBM Business Automation Configuration Container deployment.
8. Click the link in the **Node port** value to start the configuration tool.

Results

The configuration tool enables you to create and name a deployment, choose which container images to deploy, and provide the details of your supporting software to deploy a functional content services environment.

(V5.5.3 and earlier) Preparing for deployment on certified Kubernetes

Make sure that you install the necessary software before you go to the GitHub repositories to find further information on installing the products.

Before you begin

Starting in version V5.5.4, only operator deployment is supported. To deploy V5.5.4 or later, see "(V5.5.4 and later) Preparing for deployment with an operator" on page 25.

The following list is required before you install any of the automation containers.

- Kubernetes 1.11+.
- Helm 2.9.1, if you plan to install the Helm charts with Helm and Tiller.
- Kubernetes CLI. For more information, see https://kubernetes.io/docs/tasks/tools/install-kubectl/.
- 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 databases for your intended product installation.

Procedure

1. Install the required 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.
2. Use the download document to get the product archives for Kubernetes.

Creating volumes and folders for deployment on Kubernetes

The content services component containers require 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

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.

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

Important: Some environments require multiple Content Search Services deployments. To deploy multiple Content Search Services instances, specify a unique release name and service name, and a new set of persistent volumes and persistent volume claims. You can reuse the same persistent volume for the indexstore if you want to have multiple Content Search Services deployments that access the same set of index collections. However, it is recommended that the other persistent volumes be unique.

Procedure

- Create the folders, persistent volumes, and persistent volume claims for the Content Platform Engine container deployment:

  Note: The tables show a single value in the example volume and volume claim name because in most cases the persistent volume and persistent volume claim have the same name.

Table 12. Volumes, volume claims, and folders for Content Platform Engine

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as Seen by Container
Content Platform Engine Liberty configuration	`/configDropins/overrides`	cpe-cfgstore-pv cpe-cfgstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides`
Content Platform Engine Liberty logs	`/logs`	cpe-logstore-pv cpe-logstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/logs`
Content Platform Engine FileNet logs	`/cpefnlogstore`	cpe-fnlogstore-pv cpe-fnlogstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/FileNet`
Content Platform Engine FileNet (upgrade only)	`/bootstrap`	cpe-bootstrapstore-pv cpe-bootstrapstore-pvc	`/opt/ibm/wlp/usr/servers/defaultServer/lib/Bootstrap`
Content Platform Engine temporary working space	`/textext`	cpe-textextstore-pv cpe-textextstore-pvc	`/opt/ibm/textext`
Content Platform Engine IBM Case Manager rules	`/icmrules`	cpe-icmrulesstore-pv cpe-icmrulesstore-pvc	`/opt/ibm/icmrules`
Content Platform Engine file system-based advanced storage device or file store	`/asa`	cpe-filestore-pv cpe-filestore-pvc	`/opt/ibm/asa`

For each of the folders, set the ownership to 50001:50000:

```
chown –Rf 50001:50000 /cpecfgstore
```

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

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-cfgstore-pv
```

```
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/configDropins/overrides
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-cfgstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-cfgstore-pv
  volumeName: cpe-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 Content Platform Engine Liberty logs volume.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-logstore-pv
  volumeName: cpe-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 Content Platform Engine FileNet logs volume.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-filestore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/asa
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-filestore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-filestore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-filestore-pv
  volumeName: cpe-filestore-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 Content Platform Engine bootstrap information (upgrade only).

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-bootstrap-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/bootstrap
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-bootstrap-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-bootstrap-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-bootstrap-pv
  volumeName: cpe-bootstrap-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 Content Platform Engine temporary working space.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-textext-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/textext
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-textext-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-textext-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-textext-pv
  volumeName: cpe-textext-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 Case Manager rules.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-icmrules-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/icmrules
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-icmrules-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cpe-icmrules-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cpe-icmrules-pv
  volumeName: cpe-icmrules-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 Content Platform Engine file system-based advanced storage device or file store.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cpe-filestore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cpe/asa
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cpe-filestore-pv
```

Persistent volume claim:

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

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

Table 13. 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`

Table 13. Volumes, volume claims, and folders for IBM Content Navigator (continued)

Volume purpose	Example Folder to Create	Example Volume and Volume Claim to Create	mountPath as seen by container
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	`/icnasperastore`	icn-asperastore-pv icn-asperastore-pvc	/opt/ibm/aspera

For each of the folders, set the ownership to 50001:50000:

```
chown –Rf 50001:50000 /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
```

- Create the persistent volumes and persistent volume claims for the Content Search Services container deployment:

Table 14. Volumes, volume claims, and folders for Content Search Services

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim to Create**	**mountPath as seen by container**
Content Search services container data	`/CSS_Server_data/sslkeystore`	css-cfgstore-pv css-cfgstore-pvc	`/opt/IBM/ContentSearchServices/CSS_Server/data`
Content Search Services temporary working space	`/CSS_Server_temp`	css-tempstore-pv css-tempstore-pvc	`/opt/IBM/ContentSearchServices/CSS_Server/temp`

Table 14. Volumes, volume claims, and folders for Content Search Services (continued)

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim to Create**	**mountPath as seen by container**
Content Search Services logs	`/CSS_Server_log`	css-logstore-pv css-logstore-pvc	`/opt/IBM/ContentSearchServices/CSS_Server/log`
Content Search Services full text indexes	`/CSS_Indexes`	css-indexstore css-indexstore-pvc	`/opt/ibm/indexareas`
Content Search Services custom store	`/csscustomstore/CSS_config`	cs-icp-customstore	`/opt/IBM/ContentSearchServices/CSS_Server/config`

For each of the folders, set the ownership to 50001:50000:

```
chown –Rf 50001:50000 /csscustomstore
```

The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the Content Search Services container data.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-cfgstore-pv
  volumeName: css-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 Content Search Services temporary working space.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-tempstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-tempstore-pv
  volumeName: css-tempstore-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 Content Search Services logs.

Persistent volume:

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

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-logstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-logstore-pv
  volumeName: css-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 Content Search Services full text indexes.

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: css-indexstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/css/CSSIndex_OS1
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: css-indexstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: css-indexstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: css-indexstore-pv
  volumeName: css-indexstore-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 Content Search Services custom store.

Persistent volume:

```
apiVersion: v1
{
  "kind": "PersistentVolume",
  "apiVersion": "v1",
  "metadata": {
    "name": "css-icp-customstore-pv",
    "labels": {}
  },
  "spec": {
    "storageClassName": "css-icp-customstore",
    "capacity": {
      "storage": "1Gi"
    },
    "accessModes": [
      "ReadWriteMany"
    ],
    "persistentVolumeReclaimPolicy": "Retain",
    "nfs": {
      "path": "/mnt/nfsshare/css/customstore",
      "server": "<NFS_Server>"
    }
  }
}
```

Persistent volume claim:

```
apiVersion: v1
{
  "kind": "PersistentVolumeClaim",
  "apiVersion": "v1",
  "metadata": {
    "name": "css-icp-customstore",
    "namespace": "default"
  },
  "spec": {
    "storageClassName": "css-icp-customstore",
    "resources": {
      "requests": {
        "storage": "1Gi"
      }
    },
    "accessModes": [
      "ReadWriteMany"
    ]
  }
}
```

- Create the persistent volumes and persistent volume claims for the Content Management Interoperability Services (CMIS) container deployment:

Table 15. Volumes, volume claims, and folders for Content Management Interoperability Services (CMIS)

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim Name**	**mountPath as seen by container**
CMIS and Liberty configuration data	`/configDropins/overrides`	cmis-cfgstore-pv cmis-cfgstore-pvc	`/cmiscfgstore/cmis/configDropins/overrides`
Logs volume for IBM Content Navigator and Liberty logs	`/cmis/logs`	cmis-logstore-pv cmis-logstore-pvc	`/cmislogstore/cmis/logs`

For each of the folders, set the ownership to 50001:50000:

```
chown –Rf 50001:50000 /cmiscfgstore
```

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

Persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: cmis-cfgstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/cmis/configDropins/overrides
    server: <NFS Server>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: cmis-cfgstore-pv
```

Persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cmis-cfgstore-pvc
  namespace: <NAMESPACE>
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: cmis-cfgstore-pv
  volumeName: cmis-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 CMIS logs.

Persistent volume:

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

Persistent volume claim:

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

Preparing the configuration files

After you prepare the LDAP, database, and storage for the content services applications, you make the connection and usage information available to the container deployment process.

About this task

For both YAML and Helm chart deployments, you must create a set of XML files that provide the configuration settings for your specific environment. The XML files contain information about your LDAP provider, databases, and other relevant configuration parameters.

When you created the persistent volumes as part of the preparation steps, you also created directories or folders to hold configuration data. Review the location of the folders so that you save your created XML configuration files to the appropriate location in your volume.

Note that in addition to the information in your XML configuration files, you also supply more settings at deploy time, depending on which method of deployment you choose:

YAML deployment
Edit the values for your environment in the YAML file before you run the deployment.

Helm chart deployment
Specify the values for your environment as command variables when you run the Helm commands to deploy your containers.

The YAML file parameters or Helm command variables encompass numerous configuration settings. To be ready to provide all the appropriate values for the deployment, carefully review the deployment parameters in the related links.

Preparing for Content Platform Engine

After you prepare the LDAP, database, and storage for Content Platform Engine, you make the connection and usage information available to the container deployment process.

About this task

For both YAML and Helm chart deployments, you must create a set of XML files that provide the configuration settings for your specific environment. The XML files contain information about your LDAP provider, databases, and other relevant configuration parameters.

Samples of the required configuration XML files by type are available in the following GitHub location: https://github.com/ibm-ecm/container-samples/tree/5.5.3/CPE/configDropins/overrides

The following sample files are available:

- DB2JCCDriver.xml
- GCD.xml
- GCD_HADR.xml
- GCD_Oracle.xml
- GCD_SQLServer.xml
- OBJSTORE.xml
- OBJSTORE_HADR.xml
- OBJSTORE_Oracle.xml
- OraJDBCDriver.xml
- OBJSTORE_SQLServer.xml
- SQLJCCDriver.xml
- ldap_AD.xml
- ldap_TDS.xml

Important:

If this deployment is intended as part of moving an existing FileNet P8 domain to containers, the configuration values must match what is being used by the existing FileNet P8 domain:

- Ensure that the Content Platform Engine authentication settings (LDAP) match those for the existing deployment, including hostname, port, and bind user to access the directory service. If the newly deployed containers cannot be used to authenticate a user to log into the Administration Console for Content Platform Engine as a P8 domain administrator, the P8 domain will not be accessible.
- Do not change the JDBC data source or JDBC XA data source names for the GCD database or object store databases.

For more information about preparing for on-premises to container deployments, review the information in Chapter 4, "Preparing the environment to upgrade an existing FileNet P8 domain," on page 99.

Procedure

1. Provide configuration details about your Lightweight Directory Access Protocol (LDAP) directory server.

   You create an XML file for your LDAP type with configuration details to support the container environment, then save the file to your configuration overrides directory.

- Active Directory

  Use the sample file to create a file called `ldap_AD.xml`. Modify `ldap_AD.xml` with your LDAP host, baseDN port, bindDN, and bindPassword. If your LDAP server uses an SSL port, use that port and change sslEnabled=”true". If you have different userFilter and groupFilter values, modify these values as well.

  ```
  <server>
     <ldapRegistry id="MyAD" realm="defaultRealm"
        host="9.30.99.169"
        baseDN="OU=Engineering,OU=FileNet,DC=ibm,DC=com"
        port="389"
        ldapType="Microsoft Active Directory"
        bindDN="CN=CEAdmin,OU=Shared,OU=Engineering,OU=FileNet,DC=ibm,DC=com"
        sslEnabled="False"
        bindPassword="password">
        <activedFilters
           userFilter="(&(objectClass=person)(cn=%v))"
           groupFilter="(&(objectClass=group)(cn=%v))"
           userIdMap="person:cn"
           groupIdMap="*:cn">
        </activedFilters>
     </ldapRegistry>
  </server>
  ```

- IBM Security Directory Server

  Use the sample file to create a file called `ldap_TDS.xml`. Modify `ldap_TDS.xml` with your LDAP host, baseDN port, bindDN, and bindPassword. If your LDAP server uses an SSL port, use that port and change sslEnabled=”true". If you have different userFilter and groupFilter values, modify these values as well.

  ```
  <server>
     <ldapRegistry id="MyTDS" realm="defaultRealm"
        host="172.16.194.107"
        baseDN="dc=hqpsidcdom,dc=com"
        port="389"
        ldapType="IBM Tivoli Directory Server"
        bindDN="cn=root"
        sslEnabled="False"
        bindPassword="password">
        <idsFilters
           userFilter="(&(cn=%v)(objectclass=person))"
           groupFilter="(&(cn=%v)(|(objectclass=groupOfNames)
  (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
           userIdMap="*:cn"
           groupIdMap="*:cn"
           groupMemberIdMap="memberof:member">
        </idsFilters>
     </ldapRegistry>
  </server>
  ```

2. Save your updated `ldap_`*`TYPE`*`.xml` file to the configuration override directory that you created for Content Platform Engine, for example, `/cpecfgstore/cpe/configDropins/overrides`.
3. Provide configuration details about the databases that you created during the preparation steps for the Global Configuration Database (GCD) and the object stores.

   Use the sample files to create XML files for your database type with configuration details to support the container environment, then save the files to your configuration overrides directory. You create and save two XML files, one for the GCD, and one for the object store. Use the example files that correspond with your database type.

   - GCD on Db2

     Use the sample file to create a file called `GCD.xml`. Modify `GCD.xml` with values for your database serverName, GCD databaseName, portNumber, user, and password.

     ```
     <server>
        <dataSource id="GCDDS" jndiName="GCDDS"
           isolationLevel="TRANSACTION_READ_COMMITTED"
           type="javax.sql.DataSource">
           <jdbcDriver libraryRef="DB2JCC4Lib"/>
           <properties.db2.jcc databaseName="GCD"
     ```

```
         serverName="9.30.210.76"
         portNumber="50000"
         user="db2inst1"
         password="password"
         resultSetHoldability="HOLD_CURSORS_OVER_COMMIT"/>
      <connectionManager maxIdleTime="1m" maxPoolSize="50"
         minPoolSize="0" reapTime="2m" enableSharingForDirectLookups="false"/>
   </dataSource>
   <dataSource id="GCDDSXA" jndiName="GCDDSXA"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.XADataSource" supplementalJDBCTrace="true">
      <properties.db2.jcc databaseName="GCD"
         serverName="9.30.210.76"
         portNumber="50000"
         user="db2inst1"
         password="password"/>
      <connectionManager maxIdleTime="1m" maxPoolSize="50"
         minPoolSize="0" reapTime="2m" enableSharingForDirectLookups="true"/>
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
   </dataSource>
</server>
```

- Object store on Db2

  Use the sample file to create a file called `OBJSTORE.xml`. Modify `OBJSTORE.xml` with values for your database serverName, object store databaseName, portNumber, user, and password.

```
<server>
   <dataSource id="Daph1DS" jndiName="Daph1DS"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.DataSource">
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
      <properties.db2.jcc databaseName="Daph1DB"
         serverName="9.30.210.76"
         portNumber="50000"
         user="db2inst1"
         password="password"
         resultSetHoldability="HOLD_CURSORS_OVER_COMMIT"/>
      <connectionManager maxIdleTime="1m" maxPoolSize="50"
         minPoolSize="0" reapTime="2m" enableSharingForDirectLookups="false"/>
   </dataSource>
   <dataSource id="Daph1DSXA" jndiName="Daph1DSXA"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.XADataSource" supplementalJDBCTrace="true">
      <properties.db2.jcc databaseName="Daph1DB"
         serverName="9.30.210.76"
         portNumber="50000"
         user="db2inst1"
         password="password"/>
      <connectionManager maxIdleTime="1m" maxPoolSize="50"
         minPoolSize="0" reapTime="2m" enableSharingForDirectLookups="true"/>
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
   </dataSource>
</server>
```

  If you want to create a second object store, copy the `OBJSTORE.xml` file, save it with the name `OBJSTORE2.xml`, and update the values for jndiName, jndiXAName, object store databaseName, portNumber, user, and password.

- GCD on Db2 HADR

  Use the sample file to create a file called `GCD_HADR.xml`. Modify `GCD_HADR.xml` with values for your database serverName, GCD databaseName, portNumber, user, password, clientRerouteAlternateServerName, and clientRerouteAlternatePortNumber.

```
<server>
   <dataSource id="GCDDS" jndiName="GCDDS"
      validationTimeout="15"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.DataSource">
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
      <properties.db2.jcc databaseName="GCD"
         serverName="9.30.210.76"
         portNumber="50000"
         user="db2inst1"
         password="password"
         resultSetHoldability="HOLD_CURSORS_OVER_COMMIT"
         clientRerouteAlternateServerName="oclv0462.ibm.com"
         clientRerouteAlternatePortNumber="50000"
         retryIntervalForClientReroute="3"
         maxRetriesForClientReroute="15"/>
      <connectionManager maxIdleTime="1m" maxPoolSize="50"
         minPoolSize="0" reapTime="2m" enableSharingForDirectLookups="false"/>
   </dataSource>
   <dataSource id="GCDDSXA" jndiName="GCDDSXA"
      validationTimeout="15"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.XADataSource" supplementalJDBCTrace="true">
      <properties.db2.jcc databaseName="GCD"
         serverName="9.30.210.76"
         portNumber="50000"
         user="db2inst1"
         password="password"
         clientRerouteAlternateServerName="oclv0462.ibm.com"
         clientRerouteAlternatePortNumber="50000"
         retryIntervalForClientReroute="3"
         maxRetriesForClientReroute="15"/>
      <connectionManager maxIdleTime="1m" maxPoolSize="50"
         minPoolSize="0" reapTime="2m" enableSharingForDirectLookups="true"/>
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
   </dataSource>
</server>
```

- Object store on Db2 HADR

  Use the sample file to create a file called `OBJSTORE_HADR.xml`. Modify `OBJSTORE_HADR.xml` with values for your database serverName, object store databaseName, portNumber, user, password, clientRerouteAlternateServerName, and clientRerouteAlternatePortNumber.

```
<server>
   <dataSource id="Daph1DS" jndiName="Daph1DS"
      validationTimeout="3"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.DataSource">
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
      <properties.db2.jcc databaseName="Daph1DB"
         serverName="9.30.210.76"
         portNumber="50000"
         user="db2inst1"
         password="password"
         resultSetHoldability="HOLD_CURSORS_OVER_COMMIT"
         clientRerouteAlternateServerName="oclv0462.ibm.com"
         clientRerouteAlternatePortNumber="50000"
         retryIntervalForClientReroute="3"
         maxRetriesForClientReroute="3"/>
      <connectionManager maxIdleTime="1m" maxPoolSize="50"
         minPoolSize="0" reapTime="2m" enableSharingForDirectLookups="false"/>
   </dataSource>
   <dataSource id="Daph1DSXA" jndiName="Daph1DSXA"
      validationTimeout="3"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.XADataSource" supplementalJDBCTrace="true">
      <properties.db2.jcc databaseName="Daph1DB"
         serverName="9.30.210.76"
         portNumber="50000"
         user="db2inst1"
         password="password"
         clientRerouteAlternateServerName="oclv0462.ibm.com"
         clientRerouteAlternatePortNumber="50000"
         retryIntervalForClientReroute="3"
         maxRetriesForClientReroute="3"/>
      <connectionManager maxIdleTime="1m" maxPoolSize="50"
         minPoolSize="0" reapTime="2m" enableSharingForDirectLookups="true"/>
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
   </dataSource>
</server>
```

  If you want to create a second object store, copy the `OBJSTORE_HADR.xml` file, save it with the name `OBJSTORE2_HADR.xml`, and update the values for jndiName, jndiXAName, object store databaseName, portNumber, user, password, clientRerouteAlternateServerName, and clientRerouteAlternatePortNumber.

- GCD on Oracle

Use the sample file to create a file called `GCD_Oracle.xml`. Modify `GCD_Oracle.xml` with values for your database URL, user, and password.

```
<server>
   <dataSource id="GCDDS" jndiName="GCDDS"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.DataSource">
      <jdbcDriver libraryRef="OracleLib"/>
      <properties.oracle URL="jdbc:oracle:thin:@//ecmOracle1.ibm.com:1521/orcl"
         user="GCDUSER"
         password="password"/>
      <connectionManager  enableSharingForDirectLookups="false"/>
      <!-- connectionManager globalConnectionTypeOverride="unshared" / -->
   </dataSource>
   <dataSource id="GCDDSXA" jndiName="GCDDSXA"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.XADataSource" supplementalJDBCTrace="true">
      <properties.oracle URL="jdbc:oracle:thin:@//ecmOracle1.ibm.com:1521/orcl"
         user="GCDUSER"
         password="password"/>
      <connectionManager  enableSharingForDirectLookups="false"/>
      <!-- connectionManager globalConnectionTypeOverride="unshared" / -->
      <jdbcDriver libraryRef="OracleLib"/>
   </dataSource>
</server>
```

- Object store on Oracle

  Use the sample file to create a file called `OBJSTORE_Oracle.xml`. Modify `OBJSTORE_Oracle.xml` with values for your object store database JDBC URL, user, and password.

```
<server>
   <dataSource id="Daph1DS" jndiName="Daph1DS"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.DataSource">
      <jdbcDriver libraryRef="OracleLib"/>
      <properties.oracle URL="jdbc:oracle:thin:@//ecmOracle1.ibm.com:1521/orcl"
         user="OSDBUSER"
         password="password"
         resultSetHoldability="HOLD_CURSORS_OVER_COMMIT"/>
      <connectionManager  enableSharingForDirectLookups="false"/>
   </dataSource>
   <dataSource id="Daph1DSXA" jndiName="Daph1DSXA"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.XADataSource" supplementalJDBCTrace="true">
      <properties.oracle URL=" jdbc:oracle:thin:@//ecmOracle1.ibm.com:1521/orcl"
         user="OSDBUSER"
         password="password"/>
      <connectionManager  enableSharingForDirectLookups="true"/>
      <jdbcDriver libraryRef="OracleLib"/>
   </dataSource>
</server>
```

  If you want to create a second object store, copy the `OBJSTORE_Oracle.xml` file, save it with the name `OBJSTORE2_Oracle.xml`, and update the values for jndiName, jndiXAName, database URL, user, and password.

- GCD on SQL Server

  Use the sample file to create a file called `GCD_SQLSever.xml`. Modify `GCD_SQLServer.xml` with values for your database URL, user, and password.

```
<server>
    <dataSource id="FNGCDDS" jndiName="FNGCDDS"
isolationLevel="TRANSACTION_READ_COMMITTED"  type="javax.sql.DataSource">
        <jdbcDriver libraryRef="3ptLibrary"/>
        <properties.microsoft.sqlserver
                        serverName="<hostname>"
                        portNumber="1433"
                        databaseName="GCDDB"
                        user="sa"
                        password="password"/>
        <connectionManager enableSharingForDirectLookups="false"/>

    </dataSource>
    <dataSource id="FNGCDDSXA" jndiName="FNGCDDSXA" 
isolationLevel="TRANSACTION_READ_COMMITTED"  type="javax.sql.XADataSource" 
supplementalJDBCTrace="true">
        <properties.microsoft.sqlserver
                        serverName="<hostname>"
                        portNumber="1433"
                        databaseName="GCDDB"
                        user="sa"
                        password="password"/>
        <connectionManager enableSharingForDirectLookups="true" />
        <jdbcDriver libraryRef="3ptLibrary"/>

    </dataSource>
</server>
```

Tip: You can enable extended character support by adding the following parameter and setting for every datasource:

```
sendStringParametersAsUnicode="true"
```

- Object store on SQL Server

  Use the sample file to create a file called `OBJSTORE_SQLServer.xml`. Modify `OBJSTORE_SQLServer.xml` with values for your object store database JDBC URL, user, and password.

```
<server>
    <dataSource id="icnDs" isolationLevel="TRANSACTION_READ_COMMITTED" jndiName="jdbc/
CIWEBDS">
        <jdbcDriver libraryRef="3ptLibrary"/>
        <properties.microsoft.sqlserver
                        serverName="<hostname>"
                        portNumber="1433"
                        databaseName="OSDB"
                        user="sa"
                        password="password"/>
    </dataSource>
    <dataSource id="icnDsXA" jndiName="icnDsXA" 
isolationLevel="TRANSACTION_READ_COMMITTED"  type="javax.sql.XADataSource" 
supplementalJDBCTrace="true">
        <properties.microsoft.sqlserver
        serverName="<hostname>"
                        portNumber="1433"
                        databaseName="OSDB"
                        user="sa"
                        password="password"/>
                <connectionManager enableSharingForDirectLookups="true" />
        <jdbcDriver libraryRef="3ptLibrary"/>
    </dataSource>
</server>
```

  If you want to create a second object store, copy the `OBJSTORE_SQLServer.xml` file, save it with the name `OBJSTORE2_SQLServer.xml`, and update the values for jndiName, jndiXAName, database URL, user, and password.

  Tip: You can enable extended character support by adding the following parameter and setting for every datasource:

```
sendStringParametersAsUnicode="true"
```

4. Save your updated `GCD_`*`TYPE`*`.xml` and `OBJSTORE_`*`TYPE`*`.xml` files to the configuration override directory that you created for Content Platform Engine, for example, `/cpecfgstore/cpe/configDropins/overrides`.
5. Copy the corresponding database driver XML files from the samples directory to the configuration override directory that you created for Content Platform Engine, for example, `/cpecfgstore/cpe/configDropins/overrides`:
   - For Db2 and Db2_HADR: `DB2JCCDriver.xml`
   - For Oracle: `OraJDBCDriver.xml`
   - For SQL: `SQLJCCDriver.xml`

6. Move a copy of your database driver JAR files to the configuration override directory that you created for Content Platform Engine, for example, `/cpecfgstore/cpe/configDropins/overrides`.
    - For Db2, copy the corresponding database driver JAR files, `db2jcc4.jar` and `db2jcc_license_cu.jar`, from the database server.
    - For the Oracle JDBC driver jar file, browse to the web page for the Oracle Database version that you are using and download the JDBC driver that supports Java™ 1.8.
    - For Microsoft SQL Server, go to the Microsoft Support website and find the version of the Microsoft JDBC Driver for SQL Server that matches your version of Microsoft SQL Server and is supported by the Content Platform Engine as specified in the IBM Software Product Compatibility Reports.
7. If you need to use your own SSL keystore and certificate files in place of the default files that are provided with the Content Platform Engine container environment, you must add the files `keystore.jks` and `truststore.jks` to the configuration override directory that you created for Content Platform Engine, for example, `/cpecfgstore/cpe/configDropins/overrides`.

Preparing for IBM Content Navigator

After you prepare the LDAP, database, and storage for IBM Content Navigator, you make the connection and usage information available to the container deployment process.

About this task

For both YAML and Helm chart deployments, you must create a set of XML files that provide the configuration settings for your specific environment. The XML files contain information about your LDAP provider, databases, and other relevant configuration parameters.

When you created the persistent volumes as part of the preparation steps, you also created directories or folders to hold configuration data. Review the location of the folders so that you save your created XML configuration files to the appropriate location in your volume.

Note that in addition to the information in your XML configuration files, you also supply more settings at deployment time, depending on which method of deployment you choose:

YAML deployment
Edit the values for your environment in the YAML file before you run the deployment.

Helm chart deployment
Specify the values for your environment as command variables when you run the Helm commands to deploy your containers.

The YAML file parameters or Helm command variables encompass numerous configuration settings. To be ready to provide all the appropriate values for the deployment, carefully review the deployment parameters in the related links.

Samples of the required configuration XML files by type are available in the following GitHub location: https://github.com/ibm-ecm/container-samples/tree/5.5.3/ICN/configDropins/overrides

The following sample files are available:

- DB2JCCDriver.xml
- ICNDS.xml
- ICNDS_HADR.xml
- ICNDS_Oracle.xml
- OraJDBCDriver.xml
- ICNDS_SQLServer.xml
- SQLJCCDriver.xml
- ldap_AD.xml
- ldap_TDS.xml

Procedure

1. Provide configuration details about your Lightweight Directory Access Protocol (LDAP) directory server.

   You create an XML file for your LDAP type with configuration details to support the container environment, then save the file to your configuration overrides directory.

   - Active Directory

     Use the sample file to create a file called `ldap_AD.xml`. Modify `ldap_AD.xml` with your LDAP host, baseDN port, bindDN, and bindPassword. If your LDAP server uses an SSL port, use that port and change sslEnabled=”true". If you have different userFilter and groupFilter values, modify these values as well.

     ```
     <server>
        <ldapRegistry id="MyAD" realm="defaultRealm"
           host="9.30.99.169"
           baseDN="OU=Engineering,OU=FileNet,DC=ibm,DC=com"
           port="389"
           ldapType="Microsoft Active Directory"
           bindDN="CN=CEAdmin,OU=Shared,OU=Engineering,OU=FileNet,DC=ibm,DC=com"
           sslEnabled="False"
           bindPassword="password">
           <activedFilters
              userFilter="(&(objectClass=person)(cn=%v))"
              groupFilter="(&(objectClass=group)(cn=%v))"
              userIdMap="person:cn"
              groupIdMap="*:cn">
           </activedFilters>
        </ldapRegistry>
     </server>
     ```

   - IBM Security Directory Server

     Use the sample file to create a file called `ldap_TDS.xml`. Modify `ldap_TDS.xml` with your LDAP host, baseDN port, bindDN, and bindPassword. If your LDAP server uses an SSL port, use that port and change sslEnabled=”true". If you have different userFilter and groupFilter values, modify these values as well.

     ```
     <server>
        <ldapRegistry id="MyTDS" realm="defaultRealm"
           host="172.16.194.107"
           baseDN="dc=hqpsidcdom,dc=com"
           port="389"
           ldapType="IBM Tivoli Directory Server"
           bindDN="cn=root"
           sslEnabled="False"
           bindPassword="password">
           <idsFilters
              userFilter="(&(cn=%v)(objectclass=person))"
              groupFilter="(&(cn=%v)(|(objectclass=groupOfNames)
     (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
              userIdMap="*:cn"
              groupIdMap="*:cn"
              groupMemberIdMap="memberof:member">
           </idsFilters>
        </ldapRegistry>
     </server>
     ```

2. Save your updated `ldap_`*`TYPE`*`.xml` file to the configuration override directory that you created for IBM Content Navigator, for example, `/icncfgstore/icn/configDropins/overrides`.
3. Provide configuration details about the data sources that you created during the preparation steps for IBM Content Navigator.

   You create XML files for your database type with configuration details to support the container environment, then save the files to your configuration overrides directory.

   - Db2

     Create a file called `ICNDS.xml`. Modify `ICNDS.xml` with values for your database serverName, portNumber, user, and password.

     ```
     <server>
        <dataSource id="ECMClientDS" jndiName="ECMClientDS"
     ```

```
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.DataSource">
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
      <properties.db2.jcc databaseName="ICNDB"
         serverName="9.30.146.152"
         portNumber="50000"
         user="db2inst1"
         password="password"
         resultSetHoldability="HOLD_CURSORS_OVER_COMMIT"/>
      <!-- connectionManager globalConnectionTypeOverride="unshared" / -->
   </dataSource>
</server>
```

- Db2 HADR

  Create a file called `ICNDS_HADR.xml`. Modify `ICNDS_HADR.xml` with values for your database serverName, portNumber, user, password, clientRerouteAlternateServerName, and clientRerouteAlternatePortNumber.

```
<server>
   <dataSource id="ECMClientDS" jndiName="ECMClientDS"
      validationTimeout="3"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.DataSource">
      <jdbcDriver libraryRef="DB2JCC4Lib"/>
      <properties.db2.jcc databaseName="ICNDB"
         serverName="9.30.146.152"
         portNumber="50000"
         user="db2inst1"
         password="password"
         resultSetHoldability="HOLD_CURSORS_OVER_COMMIT"
         clientRerouteAlternateServerName="oclv0462.ibm.com"
         clientRerouteAlternatePortNumber="50000"
         retryIntervalForClientReroute="3"
         maxRetriesForClientReroute="3"/>
      <connectionManager globalConnectionTypeOverride="unshared"
         maxPoolSize="0" purgePolicy="FailingConnectionOnly" reapTime="30s"/>
   </dataSource>
</server>
```

- Oracle

  Create a file called `ICNDS_Oracle.xml`. Modify `ICNDS_Oracle.xml` with values for your database URL, user, and password.

```
<server>
   <dataSource id="ECMClientDS" jndiName="ECMClientDS"
      isolationLevel="TRANSACTION_READ_COMMITTED"
      type="javax.sql.DataSource">
      <jdbcDriver libraryRef="OracleLib"/>
      <properties.oracle URL="jdbc:oracle:thin:@//ecmOracle1.ibm.com:1521/orcl"
         user="db2inst1"
         password="{base64}ZGIyaW5zdDE="
         resultSetHoldability="HOLD_CURSORS_OVER_COMMIT"/>
      <!-- connectionManager globalConnectionTypeOverride="unshared" / -->
   </dataSource>
</server>
```

- SQL Server

  Create a file called `ICNDS_SQLServer.xml`. Modify `ICNDS_SQLServer.xml` with values for your database URL, user, and password.

```
<server>
        <dataSource id="ECMClientDS" jndiName="ECMClientDS"
isolationLevel="TRANSACTION_READ_COMMITTED"  type="javax.sql.DataSource">
                <jdbcDriver libraryRef="3ptLibrary"/>
                <properties.microsoft.sqlserver
                        serverName="<hostname>"
                        portNumber="1433"
                        databaseName="ICNDB"
                        user="sa"
                        password="password"/>
                />
                <connectionManager enableSharingForDirectLookups="false" />
        </dataSource>
</server>
```

4. Save your updated `ICNDS_`*`TYPE`*`.xml` files to the configuration override directory that you created for IBM Content Navigator, for example, `/icncfgstore/icn/configDropins/overrides`.
5. Copy the corresponding database driver XML files from the database server to the configuration override directory that you created for IBM Content Navigator, for example, `/icncfgstore/icn/configDropins/overrides`:
   - For Db2 and Db2_HADR: `DB2JCCDriver.xml`
   - For Oracle: `OraJDBCDriver.xml`
   - For SQL Server: `SQLJCCDriver.xml`
6. Move a copy of your database driver JAR files to the configuration override directory that you created for IBM Content Navigator, for example, `/icncfgstore/icn/configDropins/overrides`.
   - For Db2, copy the corresponding database driver JAR files, `db2jcc4.jar` and `db2jcc_license_cu.jar`, from the database server.
   - For the Oracle JDBC driver jar file, browse to the web page for the Oracle Database version that you are using and download the JDBC driver that supports Java 1.8.
   - For Microsoft SQL Server, go to the Microsoft Support website and find the version of the Microsoft JDBC Driver for SQL Server that matches your version of Microsoft SQL Server and is supported by the Content Platform Engine as specified in the IBM Software Product Compatibility Reports.

Preparing for Content Search Services security
Before you deploy the Content Search Services container, you prepare the self-signed server store.

About this task

A sample of the required server store file is available in the following git hub location: https://github.com/ibm-ecm/container-samples/tree/5.5.3/CSS/CSS_Server_data/sslkeystore.

Procedure

From the git hub sample directory, download the `cssSelfsignedServerStore` file and save it to the server data folder that you created for CSS, for example, `/csscfgstore/css/CSS_Server_data/sslkeystore`.

Preparing for Content Management Interoperability Services
After you prepare the LDAP for Content Management Interoperability Services, you make the connection and usage information available to the container deployment process.

About this task

For both YAML and Helm chart deployments, you must create an XML file that provides the configuration settings for your specific environment. The XML files contain information about your LDAP provider.

Samples of the required configuration XML files by type are available in the following GitHub location: https://github.com/ibm-ecm/container-samples/tree/5.5.3/CMIS/configDropins/overrides

The following sample files are available:

- ldap_AD.xml
- ldap_TDS.xml

Procedure

1. Provide configuration details about your Lightweight Directory Access Protocol (LDAP) directory server.

   You create an XML file for your LDAP type with configuration details to support the container environment, then save the file to your configuration overrides directory.

- Active Directory

  Create a file called `ldap_AD.xml`. Modify `ldap_AD.xml` with your LDAP host, baseDN port, bindDN, and bindPassword. If your LDAP server uses an SSL port, use that port and change sslEnabled=”true". If you have different userFilter and groupFilter values, modify these values as well.

  ```
  <server>
     <ldapRegistry id="MyAD" realm="defaultRealm"
        host="9.30.99.169"
        baseDN="OU=Engineering,OU=FileNet,DC=ibm,DC=com"
        port="389"
        ldapType="Microsoft Active Directory"
        bindDN="CN=CEAdmin,OU=Shared,OU=Engineering,OU=FileNet,DC=ibm,DC=com"
        sslEnabled="False"
        bindPassword="password">
        <activedFilters
           userFilter="(&(objectClass=person)(cn=%v))"
           groupFilter="(&(objectClass=group)(cn=%v))"
           userIdMap="person:cn"
           groupIdMap="*:cn">
        </activedFilters>
     </ldapRegistry>
  </server>
  ```

- IBM Security Directory Server

  Create a file called `ldap_TDS.xml`. Modify `Ldap_TDS.xml` with your LDAP host, baseDN port, bindDN, and bindPassword. If your LDAP server uses an SSL port, use that port and change sslEnabled=”true". If you have different userFilter and groupFilter values, modify these values as well.

  ```
  <server>
     <ldapRegistry id="MyTDS" realm="defaultRealm"
        host="172.16.194.107"
        baseDN="dc=hqpsidcdom,dc=com"
        port="389"
        ldapType="IBM Tivoli Directory Server"
        bindDN="cn=root"
        sslEnabled="False"
        bindPassword="password">
        <idsFilters
           userFilter="(&(cn=%v)(objectclass=person))"
           groupFilter="(&(cn=%v)(|(objectclass=groupOfNames)
  (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))"
           userIdMap="*:cn"
           groupIdMap="*:cn"
           groupMemberIdMap="memberof:member">
        </idsFilters>
     </ldapRegistry>
  </server>
  ```

2. Save your updated `ldap_`*`TYPE`*`.xml` file to the configuration override directory that you created for Content Management Interoperability Services, for example, `/cmiscfgstore/cmis/configDropins/overrides`.
3. Download the Content Management Interoperability Services product deployment YAML file, `cmis-deploy.yml`.
4. Modify the `cmis-deploy.yml` file to update the parameter values to match your environment requirements.
   a) Update the image name parameter to match the image name that you downloaded.

      For example: `image:mycluster.cmis:8500/default/cmis:latest`

   b) Update the values for the persistent volume claims and folders that you created during your preparation steps.

   c) Update the CE_URL value with the URL for the Content Platform Engine, for example, `http://<CE_IP>:<PORT>/wsi/FNCEWS40MTOM`.

   d) If necessary, modify the default required parameters for CMIS:

```
CMC_TIME_TO_LIVE: 3600000
CRC_TIME_TO_LIVE: 3600000
USER_CACHE_TIME_TO_LIVE: 28800000
CHECKOUT_COPYCONTENT: true
DEFAULTMAXITEMS: 25
CVL_CACHE: true
SECUREMETADATACACHE: false
FILTERHIDDENPROPERTIES: true
QUERYTIMELIMIT: 180
RESUMABLEQUERIESFORREST: true
ESCAPEUNSAFESTRINGCHARACTERS: false
MAXSOAPSIZE: 180
PRINTFULLSTACKTRACE: false
FOLDERFIRSTSEARCH: false
IGNOREROOTDOCUMENTS: false
SUPPORTINGTYPEMUTABILITY: false
```

e) If necessary, adjust the settings for the minimum required JVM heap.

The file includes the following default settings:

```
JVM_HEAP_XMS: 512m
JVM_HEAP_XMS: 1024m
```

f) If necessary, adjust the settings for the Kubernetes resources.

The file includes the following default settings:

```
CPU_REQUEST: "500m"
CPU_LIMIT: "1"
MEMORY_REQUEST: "512Mi"
MEMORY_LIMIT: "1024Mi"
REPLICAS: 1
```

For more information on these resource settings, see the Kubernetes documentation.

Preparing the YAML files

If you are using YAML files to deploy your container images, you must update the appropriate YAML file with the values for your environment. Skip this task if you are using Helm charts to deploy your images.

About this task

After you set up your supporting applications and configuration files, you must update the provided YAML deployment files with the settings for your environment.

The YAML file parameters or Helm command variables encompass numerous configuration settings. To be ready to provide all the appropriate values for the deployment, carefully review the deployment parameters in the related links.

Preparing the Content Platform Engine YAML file
Before you can deploy your container images, you update the sample YAML file for Content Platform Engine deployment with the configuration settings that you created for your content services environment.

Procedure

1. Download the Content Platform Engine product deployment YAML file, `cpe-deploy.yml`.
2. Modify the `cpe-deploy.yml` file to update the parameter values to match your environment requirements:
   a) Update the image name parameter to match the image name that you downloaded.
      For example: `image:mycluster.icp:8500/default/cpe:latest`
   b) Update the values for the persistent volume claims and folders that you created during your preparation steps.
   c) If necessary, adjust the settings for the minimum required JVM heap.
      The file includes the following default settings:

```
JVM_HEAP_XMS: 512m
JVM_HEAP_XMS: 1024m
```

d) If necessary, adjust the settings for the Kubernetes resources.

The file includes the following default settings:

```
CPU_REQUEST: "500m”"
CPU_LIMIT: "1"
MEMORY_REQUEST: "512Mi"
MEMORY_LIMIT: "1024Mi"
REPLICAS: 1
```

The default settings are suggestions only. Choose settings that are appropriate for your environment. For more information on these resource settings, see the Kubernetes documentation.

Preparing the IBM Content Navigator YAML file
If you are using YAML files to deploy your IBM Content Navigator container image, you must update the appropriate YAML file with the values for your environment. Skip this task if you are using Helm charts to deploy your image.

About this task

After you set up your supporting applications and configuration files, you must update the provided YAML deployment file with the settings for your environment.

The YAML file parameters or Helm command variables encompass numerous configuration settings. To be ready to provide all the appropriate values for the deployment, carefully review the deployment parameters in the related links.

Procedure

1. Download the IBM Content Navigator product deployment YAML file, `icn-deploy.yml`.
2. Modify the `icn-deploy.yml` file to update the parameter values to match your environment requirements.
   a) Update the image name parameter to match the image name that you downloaded.

      For example: `image:mycluster.icp:8500/default/icn:latest`

   b) Update the values for the persistent volume claims and folders that you created during your preparation steps.

   c) If necessary, adjust the settings for the minimum required JVM heap.

      The file includes the following default settings:

```
JVM_HEAP_XMS: 512m
JVM_HEAP_XMS: 1024m
```

   d) If necessary, adjust the settings for the Kubernetes resources.

      The file includes the following default settings:

```
CPU_REQUEST: "500m"
CPU_LIMIT: "1"
MEMORY_REQUEST: "512Mi"
MEMORY_LIMIT: "1024Mi"
REPLICAS: 1
```

   For more information on these resource settings, see the Kubernetes documentation.

Preparing the Content Search Services YAML file

Before you can deploy your container images, you update the sample YAML file for Content Search Services deployment with the configuration settings that you created for your content services environment.

Procedure

1. Download the Content Search Services product deployment YAML file, `css-search-deploy.yml`.
2. Modify the `css-search-deploy.yml` file to update the parameter values to match your environment requirements.
   a) Update the image name parameter to match the image name that you downloaded.

      For example: `image:mycluster.icp:8500/default/css:latest`

   b) Update the values for the persistent volume claims and folders that you created during your preparation steps.

   c) If necessary, adjust the settings for the minimum required JVM heap.

      The file includes the following default setting:

      ```
      JVM_HEAP_XMX: 3072m
      ```

   d) If necessary, adjust the settings for the Kubernetes resources.

      The file includes the following default settings:

      ```
      CPU_REQUEST: "500m"
      CPU_LIMIT: "1"
      MEMORY_REQUEST: "512Mi"
      MEMORY_LIMIT: "1024Mi"
      REPLICAS: 1
      ```

      For more information on these resource settings, see the Kubernetes documentation.

Preparing the Content Management Interoperability Services YAML file

Before you can deploy your container images, you update the sample YAML file for Content Management Interoperability Services deployment with the configuration settings that you created for your CMIS environment.

Procedure

1. Download the Content Management Interoperability Services product deployment YAML file, `cmis-deploy.yml`.
2. Modify the `cmis-deploy.yml` file to update the parameter values to match your environment requirements.
   a) Update the image name parameter to match the image name that you downloaded.

      For example: `image:mycluster.cmis:8500/default/cmis:latest`

   b) Update the values for the persistent volume claims and folders that you created during your preparation steps.

   c) Update the CE_URL value with the URL for the Content Platform Engine, for example, `http://<CE_IP>:<PORT>/wsi/FNCEWS40MTOM`.

   d) If necessary, modify the default required parameters for CMIS:

      ```
      CMC_TIME_TO_LIVE: 3600000
      CRC_TIME_TO_LIVE: 3600000
      USER_CACHE_TIME_TO_LIVE: 28800000
      CHECKOUT_COPYCONTENT: true
      DEFAULTMAXITEMS: 25
      CVL_CACHE: true
      SECUREMETADATACACHE: false
      FILTERHIDDENPROPERTIES: true
      QUERYTIMELIMIT: 180
      RESUMABLEQUERIESFORREST: true
      ESCAPEUNSAFESTRINGCHARACTERS: false
      ```

```
MAXSOAPSIZE: 180
PRINTFULLSTACKTRACE: false
FOLDERFIRSTSEARCH: false
IGNOREROOTDOCUMENTS: false
SUPPORTINGTYPEMUTABILITY: false
```

e) If necessary, adjust the settings for the minimum required JVM heap.

The file includes the following default settings:

```
JVM_HEAP_XMS: 512m
JVM_HEAP_XMS: 1024m
```

f) If necessary, adjust the settings for the Kubernetes resources.

The file includes the following default settings:

```
CPU_REQUEST: "500m"
CPU_LIMIT: "1"
MEMORY_REQUEST: "512Mi"
MEMORY_LIMIT: "1024Mi"
REPLICAS: 1
```

For more information on these resource settings, see the Kubernetes documentation.

Chapter 4. Preparing the environment to upgrade an existing FileNet P8 domain

The steps you take to prepare your environment vary depending on the supported platform, tools, and methodology you choose to perform your deployment. In any case, the FileNet P8 Platform containers require the installation or update of prerequisite software.

Before you begin

Check the IBM Software Product Compatibility Report for hardware and system requirements as well as the appropriate versions of supporting software.

About this task

Each component container needs access to the existing directory server, a database installation, and storage as well as new resources to support the content services containers.

This task is a part of the process to move an existing P8 domain to be managed by a container deployment of the Content Platform Engine and associated components. If your FileNet P8 domain was already upgraded by a deployment that uses WebSphere Application Server, certain preparation tasks that are described in this process are already accomplished and can be skipped as noted. Perform the remaining tasks that are relevant to the addition of containers to an existing FileNet P8 deployment.

Some tasks require input that results from other preparation tasks performed by other administrator roles. Plan to keep a collaboration record for use in other preparation and deployment activities.

Procedure

To prepare to introduce content services containers into an existing FileNet P8 system:

1. Review the security requirements for systems being upgraded as described in the topic Review Security upgrade planning considerations.
2. Prepare the database servers for the systems that are being upgraded.

   Use the following information:

   - Database administrator planning
   - Planning the IBM Content Search Services upgrade
3. Check the IBM Software Product Compatibility Report for memory, disk, and concurrent processing threads requirements.
4. Ensure you have the JDBC driver files that are needed for the database version in an accessible location in your Kubernetes environment.

   If your FileNet P8 is already upgraded to the same release version as the targeted container deployment, you might want to copy the JDBC driver files from the system that performed the upgrade.
5. As part of the plan to add containers to the existing FileNet P8 system or to upgrade a FileNet P8 system using containers, prepare the existing file servers for use with containers.

   You can do this preparation in the WebSphere Application Server environment prior to the upgrade to reduce the time the production system is unavailable during the upgrade process.

   a) Use the following information to ensure the existing file servers follow this guideline, or set up new file servers where the existing file storage areas will be copied: Preparing file servers for file storage areas.

      When you configure the account settings for the operating system user and group involved in securing file storage areas, the content services containers require that you use these specific account values:

Table 16. Account values for V5.5.1 containers

Pseudo Account	Numeric ID Must Be	Role
cp_os_user	501	Provides the running container access to the file system as the owner
cp_os_group	500	The group that contains the *cp_os_user*

Table 17. Account values for V5.5.2 and later containers

Pseudo Account	Numeric ID Must Be	Role
cp_os_user	50001	Provides the running container access to the file system as the owner
cp_os_group	50000	The group that contains the *cp_os_user*

b) If necessary, rearrange the storage or use another technique to provide a single top-level directory.

For existing file storage areas such as standard file stores and advanced storage area devices, and for index areas, the best practice is to have a single top-level directory for all file storage areas and another single top-level directory for all index areas. If this best practice setup was not followed, you must rearrange the storage or use other techniques to provide a single top-level directory.

- One single top-level directory is required for the persistent volume definition that corresponds to the Content Platform Engine file store volume for the Content Platform Engine container deployment.
- Another single top-level directory is needed for the persistent volume definition that corresponds to the index volume for content index data for the Content Search Services container deployment.

c) Check the folder ownership to ensure that the top-level directories and all of the subfolders are owned both by the user, recorded earlier as the value for *cp_os_user*, with the numerical ID of 501 (V5.5.1) or 50001 (V5.5.2 and later), and by a group with the numerical ID of 500 (V5.5.1) or 50000 (V5.5.2 and later).

This ownership enables to the Content Platform Engine and Content Search Services containers to access the file system. If the WebSphere Application Server continues to run with the containers to manage the FileNet P8 domain, you must also add the user account that represents the *cp_os_user* on the WebSphere Application Server system to the *cpe_os_group* in the Kubernetes environment.

6. Use the following information to enable the asynchronous processing dispatcher to ensure that the object stores progress to a completed or ready state as part of an upgrade: Enabling the Asynchronous processing dispatcher

   If your FileNet P8 domain is already upgraded to V5.5.1 or later, you can skip this step.

7. Depending on your environment choices, confirm that you have completed these additional prerequisite steps to support the deployment of the content services containers:

   a) “Configuring storage for the content services environment” on page 18.

   Storage is external to the P8 Platform container environment. You set up and configure storage to prepare for the container configuration and deployment.

   b) Prepare for logging and monitoring.

Content services containers provide optional logging and monitoring capabilities for troubleshooting, performance monitoring, and capacity planning that is based on monitoring data in real production environments.

c) Prepare your environment for container deployment:

- Prepare your IBM Cloud Private environment for your container deployment

  This process includes the following required preparation steps:

  - "Creating volumes and folders for deployment on IBM Cloud Private" on page 48
  - "Creating and loading the content services archive file" on page 63
  - "Deploying the IBM Business Automation Configuration Container" on page 67
- "(V5.5.3 and earlier) Preparing for deployment on certified Kubernetes" on page 68

  This process includes the following preparation steps:

  - "Creating volumes and folders for deployment on Kubernetes" on page 68
  - "Preparing the configuration files" on page 82
  - "Preparing the YAML files" on page 94

8. If the data will be upgraded by using the containers, provide the bootstrap information from the existing FileNet P8 `Engine-ws.ear` to the deployed containers.

   Important: If your FileNet P8 domain is already upgraded to V5.5.1 or later, you can skip this step.

   Locate and obtain a copy of the bootstrapped version of the EAR file for later use. If you used the P8 Configuration Manager tool to bootstrap the EAR file, the path to the file is as follows: `Content_Platform_Engine_installation_directory\ContentEngine\tools\configure\profiles\profile_name\ear\Engine-ws.ear`

   - If you plan to use Business Automation Configuration Console tool as described in "Upgrading a P8 domain to containers by using the IBM Business Automation Configuration Container tool" on page 121, ensure that the `Engine-ws.ear` file is in a location accessible for later upload by the configuration console tool.
   - If you plan to deploy the Content Platform Engine container as described in "Upgrading a P8 domain to containers by using the IBM Cloud Private command line" on page 122, note in that topic the instruction for where to place the `Engine-ws.ear` file for use by the Business Automation Configuration Console tool commands.
   - If you plan to deploy the Content Platform Engine container as described in "Upgrading a P8 domain to containers on certified Kubernetes" on page 124, use the following instructions to extract a required file from the `Engine-ws.ear` file:
     - Choose an unzip or archive tool such as unzip to process the `Engine-ws.ear` file. If needed, you can install unzip:

       RedHat Enterprise Linux

       ```
       yum install unzip
       ```

       Ubuntu

       ```
       sudo apt-get install unzip
       ```
     - Extract the `props.jar` file from the following location in `Engine-ws.ear`: `APP-INF/lib`.

       The following example uses the `unzip` command to place the file into the `/tmp` directory of the machine where the command is run:

       ```
       unzip -j Engine-ws.ear APP-INF/lib/props.jar -d /tmp
       ```
     - After extracting the `props.jar` file, place it into the directory that you created for the bootstrap persistent volume claim you will use for the Content Platform Engine container deployment.

9. Stop the IBM Content Search Services index dispatcher.
10. Stop all instances of the IBM Content Search Services servers that are part of this FileNet P8 domain.

    As the root user, navigate to `/opt/IBM/ContentSearchServices/CSS_Server/bin` and enter the following command:

    ```
    ./shutdown.sh
    ```
11. Stop and back up the Content Platform Engine version 5.2.x or 5.5.x before upgrading.
12. Make sure that your Content Platform Engine database is up and running.

Chapter 5. Deploying a new P8 domain by using containers

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 Chapter 3, “Preparing the environment for container deployments,” on page 13.

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.

Deploying an all-in-one developer environment

The all-in-one container platform installation tool for containers installs all the required software and configuration information needed to deploy FileNet P8 Platform in a container environment. This tool quickly creates functional P8 Platform environments for demonstration or non-production purposes only.

Before you begin

The container platform installation tool requires library files and service containers from a number of different locations. Before you begin, verify that you have valid login credentials for the following image sources:

- The Docker hub
- The Docker store
- The IBM Github repository
- IBM Passport Advantage

About this task

In a standard container deployment, you use your existing database and directory server installations, or create new ones, to support the content management services containers. With the container platform installation tool, however, those supporting software prerequisites are installed and configured by the tool. The result is a complete container platform that is ready to use in a very short amount of time.

Because this complete platform relies on open LDAP and a generic database configuration, this installation method is not appropriate for production-level use. You can use this container platform installation tool to create environments for demonstration or testing purposes, or to try out the P8 Platform system in a container environment before you move to containers.

Procedure

- To create an all-in-one developer or demonstration container environment, use the instructions in the following link: ECM Container Platform Installation Tool.

(V5.5.4 and later) Installing an operator on Kubernetes

You use the operator YAML manifest files in the GitHub repository to deploy an operator and use a custom resource to apply deployments of the automation containers on Kubernetes.

Before you begin

You must prepare your environment and cluster before you follow the instructions on GitHub. For more information, see Chapter 3, "Preparing the environment for container deployments," on page 13.

About this task

Instructions and sample files for deploying the containers on Kubernetes are provided in GitHub. The following deployment options are available:

Table 18. Options for deployment with the operator

Platform	**Where to go**
Managed Red Hat OpenShift on IBM Cloud Public	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/roks/README.md
Red Hat OpenShift	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/ocp/README.md
Certified Kubernetes	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/k8s/README.md

Note: After you deploy the component containers, return to "Completing post-deployment startup tasks for a new domain" on page 109 to finish configuring your environment.

(V5.5.3 and earlier) Deploying containers on IBM Cloud Private

You can use the IBM Business Automation Configuration Container to configure and deploy your containers. You can also edit the appropriate values in the YAML configuration files and deploy the containers from the command line.

About this task

You can choose different methods for deploying containers depending on your preference and your deployment strategy.

Configuration tool

The IBM Business Automation Configuration Container provides an interface for entering configuration parameters and deploying the container based on the parameters. The configuration tool also provides options for configuring logging and monitoring, initialization of your Content Platform Engine, and verification of your overall system.

If you plan to use the configuration container, you must prepare your environment and have a full set of parameter information on hand, including database and LDAP information and IBM Cloud Private resources details.

Command line interface

You can create YAML files that include all of the necessary configuration details for your environment. You can use the command line interface of the configuration tool with the YAML files to deploy your containers. You can also optionally configure logging and monitoring, initialization of your Content Platform Engine, and verification of the overall system.

If you plan to use the command line utility to configure and deploy the container, make sure you have installed the IBM Cloud Private CLI before you start the configuration steps.

Review and use the following guidelines when you add a container deployment:

Content Platform Engine container settings

- Use the persistent volume claim that was created previously for the Content Platform Engine filestore volume as the “File PVC name” parameter.
- Minimum resource requests values:
  - MEMORY_REQUEST: “512Mi”
  - CPU_REQUEST: “500m”
- Minimum resource limits values:
  - MEMORY_LIMITS: “2048Mi”
  - CPU_LIMIT: “1”

  The default values are suggestions only. Determine the values that are appropriate for your environment.

Content Search Services container settings

- Use the persistent volume claim created previously for the Index volume for content index data as the “Index PVC name” parameter.
- The number of replicas for the CSS containers can only be set to 1. If you wish to create additional CSS server instances, create a distinct CSS service for each CSS container deployment.
- Minimum resource requests values:
  - MEMORY_REQUEST: “512Mi”
  - CPU_REQUEST: “500m”
- Minimum resource limits values:
  - MEMORY_LIMITS: “8192Mi”
  - CPU_LIMIT: “8”

  The default values are suggestions only. Determine the values that are appropriate for your environment.

Deploying containers by using the configuration tool

You can use the IBM Business Automation Configuration Container to deploy your container environment. The configuration tool is deployed as a container in IBM Cloud Private.

Before you begin

This task assumes that you have already downloaded and prepared the image package, imported the package into IBM Cloud Private, and deployed the IBM Business Automation Configuration Container on your IBM Cloud Private cluster. For details, see “Deploying the IBM Business Automation Configuration Container” on page 67.

About this task

When you run the configuration tool that is provided by the IBM Business Automation Configuration Container, you supply numerous values for the required configuration parameters that define your environment. Most of these parameters result from the prerequisite steps that you performed to prepare for the container deployment. Before you start the process of deploying your containers with the configuration tool, review the related topics that you used to prepare your environment. These topics provide details that help you collect your input for the configuration tool.

You can also review the information in “(V5.5.3 and earlier) Configuration reference” on page 222.

Procedure

To deploy using the configuration container:

1. Access the configuration tool by clicking **Workloads** > **Helm release** > ***release_name*** > **Services**.

The **Node port** value provides a link that you can click to start the configuration tool.

2. Start the deployment, provide a name, and select IBM Cloud Private as the Cluster type.
3. Provide configuration details about your IBM Cloud Private environment.
4. Choose the components that you want to deploy.

   For each component you deploy, you must enter the configuration details that are requested by the configuration tool.
5. Choose whether you want to enable logging and monitoring.

   You provide details about the logging and monitoring service that you want to enable.
6. Choose whether you want the configuration tool to initialize and verify your installation.

   If so, you provide more details for the tool to use for setting up your components.
7. Enter the configuration information for the database and directory server that you configured for your container environment.

   You must also upload the required JDBC driver.
8. After you enter and review all the required configuration values, click **Deploy**.

   The deployment results page displays. The page contains three parts:

   - The progress of the deployment, whether finished or in progress. Once started, the **Action** link is enabled and you can click the link to download the detailed log file. There is also a re-run option if the task fails.
   - The service URL. After the provision task finishes, the available service URL displays
   - Components logs. You can check the real time log results.

   You can use the download icon to download the configuration files.

   For each deployment of the IBM Business Automation Configuration Container, you can only deploy the content services containers one time. After deployment, if you open the configuration tool again, you see the deployment results page.

Enabling SSL for open source container logging by using IBM Cloud Private
If you require SSL for your open source logging for containers, you can use the IBM Cloud Private console to enable SSL on your Logstash server.

About this task

These updates must be repeated for each container deployment where you have open source logging enabled.

Procedure

To enable SSL on your Logstash server:

1. In the IBM Cloud Private console, navigate to **Workloads** > **Deployments**, and find the deployment that you want to update.
2. Right-click the deployment, and click **Edit**.

   The Edit Deployment window displays the JSON file for the deployment.
3. In the JSON file, navigate through the JSON key path spec to `template` > `spec` > `containers` > `volumeMounts`.
4. Insert the following mount into the existing mount list:

   ```
   {
   "mountPath": "/etc/pki/tls/certs",
   "name": "logmon-cert-volume"
   }
   ```
5. Confirm that no JSON error results from the change, and submit the update.

   After you submit, IBM Cloud Private includes your change in a rolling upgrade in the background.
6. Repeat these steps for all of your deployments that need SSL enabled for open source logging.

Deploying containers by using the configuration command line

You can use the command line version of the IBM Business Automation configuration tool to deploy your container environment.

About this task

The command line deployment uses a `Jobs.yml` file to define the configuration of the environment.

You must review and update all the configuration values in the file before you run the deployment to ensure that your deployment succeeds.

Procedure

To deploy using the configuration command line:

1. Download and prepare the `ibacc-master-cli.yml` file:
   a) Download `ibacc-master-cli.yml` from the following location:
      `https://github.com/ibm-ecm/container-samples/tree/5.5.3/miscibacc-master-cli.yml`
   b) Edit the following values in the `ibacc-master-cli.yml` file to supply the appropriate values for your environment:

      ```
      spec.template.spec.containers[0].image:
      clustername.icp:port/default/ibacc-master:latest

      spec.template.spec.containers[0].env:
      - name: SC_ECC_DEPLOY_ID
      value: deployment id for isolation, such as ecm-staging-deploy

      spec.template.spec.containers[0].volumeMounts:ibacc-cfg-volume
      spec.template.spec.volumes: "/opt/ibm/ibacc"
      ```
2. Download templates for the YAML files that are used to deploy the content services containers:

   ```
   docker run -v /LOCAL_FOLDER:/opt/ibm/ibacc/cfg/template
   --rm mycluster.icp:8500/your-namespace/ibacc-master:latest
   bash /exportConfigFileTemplate.sh
   ```

   The templates are added to the location in the folder that you specified for the *LOCAL_FOLDER* in the command.
3. Use the templates to create the following YAML files for your deployment:
   - `Jobs.yml`
   - `IBACC-ecm-provision-icp-job.yml`
   - `IBACC-init-deploy.yml`
   - `IBACC-verify-deploy.yml`
4. Review all values in the `Jobs.yml` file and update values as needed.
   For details about the values that are needed for the deployment, see the information in "(V5.5.3 and earlier) Configuration reference" on page 222.
5. Use the following command to determine the persistent volume claim value for the configuration container:

   ```
   # kubectl get pvc -n ecm-icp-example |grep -i ecm-icp-ibacc-demo
   ```

   Where ecm-icp-example is the namespace that you created for the configuration container, and ecm-icp-ibacc-demo is the persistent volume claim for the configuration tool job container.

   The command returns results like the following example:

   ```
   ecm-icp-ibacc-demo    Bound
   pvc-82c56c78-4db6-11e8-a72b-00163e01b69b     1Gi    RWX ecm-icp-nfs     6h
   ```
6. Change to the nfs storage node by using the information that was returned in the previous command:

```
#cd /nfsstorage/pvc-82c56c78-4db6-11e8-a72b-00163e01b69b
```

7. Create a configuration directory:

   ```
   #mkdir -p SC_ECC_DEPLOY_ID/cfg
   ```

   For example:

   ```
   #mkdir -p ecm_staging_deploy/cfg
   ```

8. Copy your updated `Jobs.yml` file to your newly created configuration directory.
9. Use the provisioning command of the configuration container to deploy your container environment:

   ```
   #kubectl apply -f ibacc-master-cli.yml ***
   ```

10. When the deployment completes, check the ibacc-ICp provisioning log `ibacc.log` in the `/nfsstorage/pvc-82c56c78-4db6-11e8-a72b-00163e01b69b/ecm_staging_deploy/cfg/logs/ibacc-ICp` directory.

Enabling SSL for open source container logging by using the command line
If you are using the open source ELK stack for logging and you require SSL for the logging component, you can use the command line to enable SSL on your Logstash server.

Before you begin
The steps in this task assume that you have configured Kubectl for use.

About this task

These updates must be repeated for each container deployment where you have open source logging enabled.

Procedure

To enable SSL for open source container logging:

1. Use the following command to get all the deployment names for your content services deployments:

   ```
   kubectl get deployment -n your working namespace for ECM
   ```

   For example:

   ```
   kubectl get deployment -n ecm-icp-space
   ```

   Choose one of the returned deployments to enable.
2. Get the container name for the deployment for which you want to enable SSL logging:

   ```
   kubectl get deployment -n your working namespace for ECM
   -o wide | grep cpe deployment name | awk '{print $7}'
   ```

   For example, if your Content Platform Engine deployment name is ibacc-cpe-ibm-dba-contentservices, you run the following command

   ```
   kubectl get deployment -n ecm-icp-space
    o wide | grep ibacc-cpe-ibm-dba-contentservices  | awk '{print $7}
   ```

3. Add a patch on your deployment by adding the following lines to your file:

   ```
   kubectl patch deployment product deployment name determined by get deployment command -p \
   '{
       "spec": {
           "template": {
               "spec": {
                   "containers": [{
                       "name": "product deployment name determined by get deployment command",
                       "volumeMounts": [{
   ```

```
                        "mountPath": "/etc/pki/tls/certs",
                        "name": "logmon-cert-volume"
                    }]
                }],
                "volumes": [{
                    "name": "logmon-cert-volume",
                    "persistentVolumeClaim": {
                        "name": " logmon-cert-cfgstore"
                    }
                }]
            }
        }
    }
```

This example uses the variable returned product deployment name, *product deployment name determined by get deployment command*. You use your own product deployment name value, returned in the previous step, in place of that value.

(V5.5.3 and earlier) Deploying containers on certified Kubernetes

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

Before you begin

You must prepare your environment before you follow the instructions on GitHub. For more information, see "(V5.5.3 and earlier) Preparing for deployment on certified Kubernetes" on page 68.

About this task

Instructions and sample files for deploying content services on Kubernetes are provided in the following GitHub repository https://github.com/ibm-ecm/container-samples/tree/5.5.3.

Completing post-deployment startup tasks for a new domain

After you run the container deployment, you perform additional tasks to configure and start your P8 domain.

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 Content Platform Engine and IBM Content Navigator deployments.

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:

External share
This setting might interfere with the use of external share.

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.

Enabling custom SSL certificates for Content Search Services

The Content Platform Engine and Content Search Services images are created with sample SSL configuration and sample self-signed certification. If you need to run containers with your own existing customized certificate, this customized certificate must be imported into the related keystore files for both the Content Platform Engine container and the Content Search Services container.

About this task

This procedure uses the following sample values:

- The public and private key algorithm is RSAs.
- The key size is 2048.
- The certification signature algorithm is SHA256withRSA.

Run the keytool command with IBM Java V1.8 or above.

Procedure

To enable customized certificates for SSL communication:

1. Import the customized certificate into the Content Search Services container and into the sample Content Search Services keystore file.
   a) Use the following command:

```
keytool -importcert -alias cusalias -file cusCert.cer
 -keystore cssSelfsignedServerStore -storepass changeit
```

   Where the following example values are replaced by the values for your environment:

   - The customized certificate file is *cusCert.cer*.
   - The sample Content Search Services keystore file is *cssSelfSignedServerStore*.
   - The keystore password is *changeit*.
   - The mount path of *cssSelfSignedServerStore* is */opt/IBM/ContentSearchServices/CSS_Server/data*.

   Note: The alias value must be different from the existing ones in the keystore.

   b) Restart the Content Search Services containers.
2. Configure customized certification on the Content Platform Engine container:
   a) Create the keystores file to contain the customized certification.

   (If a keystore file already exists, skip this step.)

   Use the following command:

```
keytool -genkey -v -keyalg RSA -keypass cusPassWord -keystore csskeystore.jks
-storepass cusPassWord -validity 3650 -dname "CN=CPE, OU=ECM, O=IBM, L=Unknown,
ST=Unknown, C=Unknown"
-keysize 2048
keytool -genkey -v -keyalg RSA -keypass cusPassWord
-keystore csstruststore.jks -storepass cusPassWord -validity 3650
-dname "CN=CPE, OU=ECM, O=IBM, L=Unknown, ST=Unknown, C=Unknown" -keysize 2048
```

   b) Run the following command to import the certificate:

```
keytool -importcert -alias cusalias -file cusCert.cer -keystore csskeystore.jks
-storepass cusPassWord
```

```
keytool -importcert -alias cusalias -file cusCert.cer -keystore csstruststore.jks
-storepass cusPassWord
```

Remember: The alias value should be different than the ones that exist in the keystore.

The password can be entered in clear text or in an encoded form. The `securityUtility` encode option can be used to encode the password:

```
kubectl exec -it cpe /opt/ibm/wlp/bin/securityUtility encode cusPassWord
{xor}PCosDz4sLAgwLTs=
```

c) Move the keystore files to the host directory configuration overrides directory in the mount path of the container:

For example:

```
/home/cpe_data/configDropins/overrides/csskeystore.jks
/home/cpe_data/configDropins/overrides/csstruststore.jks
```

d) Create a Liberty SSL configuration XML file in the Liberty overrides host directory, for example, create a file called `cpe-ssl.xml` in the `/cpecfgstore/cpe/configDropins/overrides` directory.

The location values of the keystores that you specify in the XML file must use the full mount path inside the container. The following example shows the contents of the XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
    <!-- SSL keystore and truststore configuration for CSS -->
    <ssl id="cusSSLSettings"
        keyStoreRef="cusKeyStore"
        trustStoreRef="cusTrustStore"
        clientAuthenticationSupported="false"
        sslProtocol="TLSv1.2"
        securityLevel="CUSTOM"
        enabledCiphers="SSL_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_AES_128_GCM_SHA2
56 SSL_RSA_WITH_AES_128_CBC_SHA256"
        />
    <keyStore id="cusKeyStore"
        location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
csskeystore.jks"
        type="JKS" password="{xor}PCosDz4sLAgwLTs=" />
    <keyStore id="cusTrustStore"
        location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
csstruststore.jks"
        type="JKS" password="{xor}PCosDz4sLAgwLTs=" />

</server>
```

Enabling SSL between Liberty and your LDAP server

The deployment for content services containers does not automatically enable the Liberty configuration for SSL communication between Liberty and your LDAP server. If you require SSL communication, you can create this configuration manually.

About this task

The following steps assume you have already deployed the content services containers successfully on your IBM Cloud Private environment, and you need to enable SSL communication afterwards. The applicable content services product components running on Liberty are Content Platform Engine, IBM Content Navigator, and Content Management Interoperability System.

Procedure

To enable an SSL connection with the LDAP server:

- Locate the storage folder for your override LDAP configuration for Liberty running ECM products. The override LDAP xml is usually named as ldap_TDS.xml or ldap_AD.xml, which one depends on your LDAP type.

1. Locate the configuration override files for the LDAP configuration for your content services environment.

   The configuration override file for LDAP is usually named `ldap_TDS.xml` or `ldap_AD.xml`, depending on your LDAP type. The file is located in the overrides directory, for example: `/cpecfgstore/configDropins/overrides`.
2. Update the LDAP XML override file to configure entries to the required key store files.

   Refer to the Liberty documentation for details on how to create or provide the required key store files.
3. Save the updated LDAP XML file to your configuration overrides directory, for example, `/cpecfgstore/configDropins/overrides`.

   When the Liberty detects an update to the LDAP configuration file in the overrides directory, the server refreshes.
4. If you have already created the directory server configuration in the Administration Console for Content Platform Engine, update the configuration in the console to reflect your SSL configuration changes.

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 for the IBM Cloud Private environment, for example: `http://ICP_IP_Address:30557/sync/notify`.

Initializing and verifying your content services environment

If you chose not to have the configuration tool perform the initialization and verification steps for you automatically, you must manually perform the initial tasks that are required to have a working FileNet P8 domain environment.

About this task

After your components are installed, it is necessary to create additional elements in your system to create a working FileNet P8 domain and content services environment.

Procedure

1. Create the FileNet P8 domain.

   For instructions, see the following information: Creating the FileNet P8 domain
2. Create the database connection.

For instructions, see the following information: Creating a database connection

3. Create an initial object store.

   For instructions, see the following information: Creating the initial object store
4. Verify the Content Platform Engine system.

   For instructions, see the following information: Verifying the Content Platform Engine system

Optional - Additional configuration and tuning for Content Search Services

Some functions in Content Search Services require additional configuration or tuning steps in a container environment.

About this task

There is a known issue with running the following Content Search Services tools in a container environment:

- ConfigTool (`configTool.sh`)
- AdminTool (`adminTool.sh`)
- dumpIndex (`dumpIndex.sh`)

Several configuration setting changes require that you stop the Content Search Services server processes. This action is not supported in a container environment.

To make configuration changes, you can choose one of the following methods:

- In Content Search Services V5.5.3 or later, a new custom store persistent volume is available that is mapped to`/opt/IBM/ContentSearchServices/CSS_Server/config` directory inside the container. You can manually edit the `configuration.xml` file in the custom store volume mapping to make changes to the Content Search Services configuration. After the configuration changes, you must restart or recreate the Content Search Services pod in order to apply the updates.
- Contact IBM Support for assistance.

To run the AdminTool or the dumpIndex tool, contact IBM support for assistance.

Procedure

- For more information on tuning and configuration changes for Content Search Services, see Configuration tool parameters

Optional - Creating an Ingress resource for content services containers

You can create an Ingress resource to control web access to your deployed containers.

Before you begin

This task assumes that you have the nginx-ingress controller as part of your Kubernetes environment.

About this task

You create an Ingress resource that works with your Ingress controller to manage external access to the services in your cluster. The Ingress resource provides a list of rules to use when assessing incoming access requests.

The steps provided demonstrate how to create Ingress resources for the following scenarios:

- Accessing Content Platform Engine
- Accessing IBM Content Navigator
- Creating a single endpoint resource for both Content Platform Engine and IBM Content Navigator

Procedure

1. Update the configmap for your Ingress controller to enable the required feature.

Use the following commands:

IBM Cloud Private

```
kubectl get configmap -n kube-system
```

```
kubectl edit configmap nginx-ingress-controller -n kube-system
```

Certified Kubernetes

```
kubectl get configmap -n <ingress-namespace>
```

```
kubectl edit configmap nginx-ingress-controller -n <ingress-namespace>
```

Add the following setting:

```
enable-underscores-in-headers: "true"
```

2. Use the following commands to remove the existing Ingress controller pod and let it recreate:

   IBM Cloud Private

```
kubectl get pod -n kube-system
```

```
kubectl delete pod <ingress pod> -n kube-system
```

   Certified Kubernetes

```
kubectl get pod -n <ingress-namespace>
```

```
kubectl delete pod <ingress pod> -n <ingress-namespace>
```

   For example:

```
kubectl delete pod nginx-ingress-controller-l79qf -n kube-system
```

3. Configure the Ingress resource for Content Platform Engine:

   a) Create a ClusterIP service for Content Platform Engine:

```
---
apiVersion: v1
kind: Service
metadata:
 name: ibacc-cpe-ingress-svc
spec:
ports:
 -name: http
 protocol: TCP
 port: 9080
 targetPort: 9080
 -name: https
 protocol: TCP
 port: 9443
 targetPort: 9443
selector:
 app: ibm-dba-contentservices
type: ClusterIP
---
```

   b) Create an SSL certificate for Content Platform Engine:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /tmp/tls.key -out /tmp/
tls.crt -subj "/CN=cpe.icp"
```

   c) Create a secret with the SSL certificate:

```
kubectlcreate secret tls cpe --key /tmp/tls.key --cert /tmp/tls.crt
```

d) Use the secret and a virtual host to create the Ingress resource:

```
------
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: cpe-ingress
 annotations:
  kubernetes.io/ingress.class: nginx
  ingress.kubernetes.io/affinity: cookie
  ingress.kubernetes.io/backend-protocol: "HTTPS"
  ingress.kubernetes.io/secure-backends: "true"
  ingress.kubernetes.io/session-cookie-name: route
  ingress.kubernetes.io/session-cookie-hash: sha1
spec:
 rules:
 -host: cpe.icp
 http:
  paths:
   -backend:
     serviceName: ibacc-cpe-ingress-svc
     servicePort: 9443
    path: /
tls:
 -hosts:
  -cpe.icp
  secretName: cpe
---
```

e) On the system you use to access the Content Platform Engine URL, update `/etc/hosts` with the following value: `< ICP Public IP Address> cpe.icp`

You then use `https://cpe.icp/acce` to access the Administration Console for Content Platform Engine.

4. Configure the Ingress resource for IBM Content Navigator:

a) Create a ClusterIP service for IBM Content Navigator:

```
---
apiVersion: v1
kind: Service
metadata:
 name: ibacc-icn-ingress-svc
spec:
ports:
 -name: http
 protocol: TCP
 port: 9080
 targetPort: 9080
 -name: https
 protocol: TCP
 port: 9443
 targetPort: 9443
selector:
 app: ibm-dba-navigator
type: ClusterIP
sessionAffinity: ClientIP
---
```

b) Create an SSL certificate for IBM Content Navigator:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /tmp/tls.key -out /tmp/
tls.crt -subj "/CN=icn.icp"
```

c) Create a secret with the SSL certificate:

```
kubectlcreate secret tls icn --key /tmp/tls.key --cert /tmp/tls.crt
```

d) Use the secret and a virtual host to create the Ingress resource:

```
------
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: icn-ingress
 annotations:
  kubernetes.io/ingress.class: nginx
  ingress.kubernetes.io/affinity: cookie
  ingress.kubernetes.io/backend-protocol: "HTTPS"
  ingress.kubernetes.io/secure-backends: "true"
  ingress.kubernetes.io/session-cookie-name: route
  ingress.kubernetes.io/session-cookie-hash: sha1
spec:
 rules:
 -host: icn.icp
 http:
  paths:
   -backend:
     serviceName: ibacc-icn-ingress-svc
     servicePort: 9443
    path: /
tls:
 -hosts:
  -icn.icp
  secretName: icn
---
```

e) On the system you use to access the IBM Content Navigator URL, update `/etc/hosts` with the following value: `< ICP Public IP Address> cpe.icp`

You then use `https://icn.icp/navigator` to access IBM Content Navigator.

5. Use the secrets that you created previously to configure a single endpoint Ingress resource for both Content Platform Engine and IBM Content Navigator:

```
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: dbamc-ingress
 annotations:
   kubernetes.io/ingress.class: "nginx"
   # The NGINX ingress annotations contains a new prefix nginx.ingress.kubernetes.io.
   # To avoid breaking a running NGINX ingress controller, specify both new and old prefixes.
   ingress.kubernetes.io/affinity: "cookie"
   nginx.ingress.kubernetes.io/affinity: "cookie"
   ingress.kubernetes.io/session-cookie-name: "route"
   nginx.ingress.kubernetes.io/session-cookie-name: "route"
   ingress.kubernetes.io/secure-backends: "true"
   nginx.ingress.kubernetes.io/secure-backends: "true"
   ingress.kubernetes.io/backend-protocol: "HTTPS"
   nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
   ingress.kubernetes.io/session-cookie-hash: "sha1"
   nginx.ingress.kubernetes.io/session-cookie-hash: "sha1"
   ingress.kubernetes.io/use-regex: "true"
   nginx.ingress.kubernetes.io/use-regex: "true"
spec:
 rules:
 - host: dbamc.ibm.com
   http:
     paths:

     - backend:
          serviceName: ibacc-cpe-ingress-svc
          servicePort: 9443
        path: /acce(.*)
      - backend:
          serviceName: ibacc-cpe-ingress-svc
          servicePort: 9443
        path: /P8CE/(.*)
      - backend:
          serviceName: ibacc-cpe-ingress-svc
          servicePort: 9443
        path: /FileNet/(.*)
      - backend:
          serviceName: ibacc-cpe-ingress-svc
          servicePort: 9443
        path: /wsi/(.*)
      - backend:
          serviceName: ibacc-icn-ingress-svc
          servicePort: 9443
        path: /navigator/(.*)
      - backend:
          serviceName: ibacc-icn-ingress-svc
          servicePort: 9443
        path: /sync/(.*)
```

```
  tls:
  - hosts:
    - dbamc.ibm.com
    secretName: dbamc-ingress-tls
```

Chapter 6. Moving an existing P8 domain to containers

If you already have an on-premises installation at V5.5.1 or later, you can add a container deployment to your existing P8 domain. You can also choose to upgrade your environment from your existing on-premises installation level to a V5.5.1 or later container deployment.

About this task

If your current P8 domain is supported by WebSphere Application Server, you can add a container deployment or upgrade your environment to a container deployment.

Check the IBM Software Product Compatibility Report for the appropriate versions of supporting software.

Adding containers to an existing on-premises P8 domain (side-by-side deployment)

You can deploy the content services containers to transition a WebSphere Application Server deployment to an IBM Cloud Private deployment to manage an existing FileNet P8 domain. You can maintain both the on-premises environment and the container environment.

Before you begin

Although you are adding new container deployments rather than replacing your existing system, it is still necessary to prepare the existing environment in the same way that you would for an upgrade. Before you perform this task, ensure that you have reviewed and performed the tasks listed in Chapter 4, “Preparing the environment to upgrade an existing FileNet P8 domain,” on page 99.

About this task

When you add a container deployment to an existing P8 domain, you follow the same procedures as for a container deployment in a new environment. However, you use the backend support configuration and storage that already exists for your domain. Review and use the following guidelines when you add a container deployment:

Use existing LDAP server and database
Provide connection and configuration information for same the LDAP server and databases that is used by the existing WebSphere Application Server managed FileNet P8 domain. If you follow the model that described in “(Optional) Upgrade with migration to containers using a staging environment” on page 9, the connection information points to the database replicas.

Data source names
Use the data source names that are used in the existing system.

Content Platform Engine container settings

- Use the persistent volume claim that was created previously for the Content Platform Engine filestore volume as the “File PVC name” parameter.
- Minimum resource requests values:
  - MEMORY_REQUEST: “512Mi”
  - CPU_REQUEST: “500m”
- Minimum resource limits values:
  - MEMORY_LIMITS: “2048Mi”
  - CPU_LIMIT: “1”

Content Search Services container settings

- Use the persistent volume claim created previously for the Index volume for content index data as the "Index PVC name" parameter.
- The number of replicas for the CSS containers can only be set to 1. If you wish to create additional CSS server instances, create a distinct CSS service for each CSS container deployment.
- Minimum resource requests values:
  - MEMORY_REQUEST: "512Mi"
  - CPU_REQUEST: "500m"
- Minimum resource limits values:
  - MEMORY_LIMITS: "8192Mi"
  - CPU_LIMIT: "8"

Procedure

- Following the instructions for deploying containers:
  - IBM Cloud Private

    You can use the IBM Business Automation Configuration container tool, or use the associated command line to deploy your containers:

    - "Deploying containers by using the configuration tool" on page 105
    - "Deploying containers by using the configuration command line" on page 107
  - Certified Kubernetes

    Instructions and sample files for deploying content services on Kubernetes are provided in the following GitHub repository https://github.com/ibm-ecm/container-samples/tree/5.5.3.

Upgrading an existing P8 domain to a containers-only environment

You can upgrade your existing on-premises P8 domain to an environment that is fully deployed on containers. The process is similar to a new installation on containers, except that you use your existing configuration of supporting software like databases and directory servers. The upgrade to the newer version is accomplished by the deployment and startup of the component containers.

About this task

You can use the IBM Business Automation Configuration tool or the command line version of the tool to upgrade your P8 domain to a container-only environment from IBM Cloud Private.

An upgrade installation starts from a V5.2.x or V5.5.0 installation, and upgrades to V5.5.1 or later as part of the conversion.

The high-level steps in the upgrade topics provide a general roadmap for the upgrade deployment of your environment. Before you start the upgrade, read through the information on preparing your environment, make sure you have performed all of the required preparation steps, and review the steps for a new container deployment.

For a list of the preparation steps you must do to prepare for an upgrade, see Chapter 4, "Preparing the environment to upgrade an existing FileNet P8 domain," on page 99.

When you add a container deployment to an existing P8 domain, you follow the same procedures as for a container deployment in a new environment. However, you use the backend support configuration and storage that already exists for your domain. The process for upgrading to containers uses the bootstrap information in the deployed EAR file for Content Platform Engine to configure the environment rather than requiring new information for configuration. Review and use the following guidelines when you add a container deployment:

Use existing LDAP server and database

Provide connection and configuration information for same the LDAP server and databases that is used by the existing WebSphere Application Server managed FileNet P8 domain. If you follow the model that described in Upgrade with migration to a new server instance, the connection information points to the database replicas.

Data source names

Update the data source values to the names that are used in your existing system. Do not use the default values that are provided in the tool.

The values you use for the data source names for the GCD and the object stores must match the values in your existing on-premises installation. To determine the correct values, you can view the **Data Source** list under **Services** in your WebSphere Application Server Domain Structure.

Alternatively, you can determine the values for your data sources by using the Administration Console for Content Platform Engine.

Content Platform Engine container settings

- Use the persistent volume claim that was created previously for the Content Platform Engine filestore volume as the “File PVC name” parameter.
- Minimum resource requests values:
  - MEMORY_REQUEST: “512Mi”
  - CPU_REQUEST: “500m”
- Minimum resource limits values:
  - MEMORY_LIMITS: “2048Mi”
  - CPU_LIMIT: “1”

Content Search Services container settings

- Use the persistent volume claim created previously for the Index volume for content index data as the “Index PVC name” parameter.
- The number of replicas for the CSS containers can only be set to 1. If you wish to create additional CSS server instances, create a distinct CSS service for each CSS container deployment.
- Minimum resource requests values:
  - MEMORY_REQUEST: “512Mi”
  - CPU_REQUEST: “500m”
- Minimum resource limits values:
  - MEMORY_LIMITS: “8192Mi”
  - CPU_LIMIT: “8”

Upgrading a P8 domain to containers by using the IBM Business Automation Configuration Container tool

You can upgrade your existing on-premises P8 domain to an environment that is fully deployed on containers. Use the IBM Business Automation Configuration tool to upload your existing Content Platform Engine application EAR file and deploy your environment.

Before you begin

Before you start the upgrade, be sure that you have completed all the steps in the following topic: Chapter 4, “Preparing the environment to upgrade an existing FileNet P8 domain,” on page 99

Procedure

To upgrade a P8 domain to containers by using the configuration tool

1. In the IBM Business Automation Configuration tool, start a new deployment and choose **Upgrade on-premises environment to container**.
2. When prompted, upload the `Engine-ws.ear` file of your existing P8 domain into the IBM Business Automation Configuration tool.
3. Confirm or provide the following input to complete the configuration tool settings:
   - Values from the LDAP and database configuration for your existing domain
   - Values of the persistent volumes and mount paths that you created for the container deployment
   - Logging and monitoring information for the new container deployment
   - All other settings that are relevant to your environment
4. Run the deployment.

   When the containers are online and mounted, the default location for the `asa` directory in the container is `/opt/ibm/asa`.
5. After the containers are up and running, move the file stores if needed.

   If you need to move the file store location, use the following information: Moving a file storage area

Upgrading a P8 domain to containers by using the IBM Cloud Private command line

You can upgrade your existing on-premises P8 domain to an environment that is fully deployed on containers. Use the command line version of the tool to deploy the containers.

Before you begin

Before you start the upgrade, be sure that you have completed all the steps in the following topic: Chapter 4, “Preparing the environment to upgrade an existing FileNet P8 domain,” on page 99

Procedure

To upgrade an on-premises P8 domain to containers

1. Download templates for the YAML files that are used to deploy the content services containers:

   ```
   docker run -v /LOCAL_FOLDER:/opt/ibm/ibacc/cfg/template
   --rm mycluster.icp:8500/your-namespace/ibacc-master:3.0.0
   bash /exportConfigFileTemplate.sh
   ```

   The templates are added to the location in the folder that you specified for the *LOCAL_FOLDER* variable in the command.
2. Use the templates to create the following YAML files for your deployment:
   - `Jobs.yml`
   - `ibacc-provision-icp-job.yml`
3. Edit your `Jobs.yml` file for the Content Platform Engine container.

   Note: You must set the SC_ECC_DEPLOYMENT_TYPE parameter to the value for upgrade, for example:

   ```
   SC_ECC_DEPLOYMENT_TYPE:upgrade
   ```

   Look for the following parameters in the `jobs.yml` file and modify as needed for your environment:

   ```
   id: ibacc-ICp

   name: IBM FileNet Content Manager Provision Job

   # path relative to ${SC_ECC_BASE_DIR}
   relativePath: ibacc-provision-icp-job.yml

   jobType: provision

   dependencies:
   # false by default
   ```

Confirm or edit other values as follows:

- Set the value for **SC_ECC_LICENSE** to accept.
- Include values from the LDAP and database configuration for your existing domain
- Add the values of the persistent volumes and mount paths that you created for the container deployment
- Decide whether you want to enable logging and monitoring information for the new container deployment, and supply relevant parameter values
- Do not enable the optional **Initialization** setting.
- Review and edit all other settings that are relevant to your environment

4. Create a new configuration directory in the location where the pre-existing configuration container stored the jobs.yml file.

   a) Use the following command to determine the name of the persistent volume mounted for the pre-existing configuration (job) container instance:

   ```
   # kubectl get pvc -n ecm-icp-example |grep -i ecm-icp-ibacc-demo
   ```

   Where ecm-icp-example is the namespace that you created for the configuration container, and ecm-icp-ibacc-demo is the persistent volume claim for the configuration tool job container.

   The command returns results like the following example:

   ```
   ecm-icp-ibacc-demo    Bound
   pvc-82c56c78-4db6-11e8-a72b-00163e01b69b     1Gi    RWX ecm-icp-nfs     6h
   ```

   b) Change your working directory to the directory discovered in step 4a.

   If you used NFS mounted storage, the change directory command might look like the following example:

   ```
   #cd /nfsstorage/pvc-82c56c78-4db6-11e8-a72b-00163e01b69b
   ```

   c) Create a new directory in your current location for the configuration container to use:

   ```
   #mkdir -p SC_ECC_DEPLOY_ID/cfg
   ```

   For example:

   ```
   #mkdir -p ecm_staging_deploy/cfg
   ```

   d) Copy your updated `Jobs.yml` file to your newly created configuration directory.

5. Create additional directories where the pre-existing configuration container stores the working files to store needed binary files.

   a) Create the following additional directories under your configuration container working location:

   ```
   SC_ECC_DEPLOY_ID/lib/jdbc
   SC_ECC_DEPLOY_ID/lib/upgrade
   ```

   For example:

   ```
   #mkdir -p ecm_staging_deploy/lib
     #mkdir -p ecm_staging_deploy/lib/jdbc
     #mkdir -p ecm_staging_deploy/lib/upgrade
   ```

   b) Copy your database JDBC driver jar files to the …`/lib/jdbc` directory.

   c) (If your FileNet P8 domain was already upgraded to Content Platform Engine V5.5.1, you can skip this step.) Place the `Engine-ws.ear` file from your existing P8 domain in the directory that you created for the bootstrap persistent volume claim.

6. Use the provisioning command of the configuration container to deploy your container environment:

```
#kubectl apply -f ibacc-provision-icp-job.yml
```

7. When the deployment completes, check the ibacc-ICp provisioning log `ibacc.log` in your configuration container directory under `/ecm_staging_deploy/cfg/logs/ibacc-ICp`.

Upgrading a P8 domain to containers on certified Kubernetes

Before you begin

Before you start the upgrade, be sure that you have completed all the steps in the following topic: “(V5.5.3 and earlier) Preparing for deployment on certified Kubernetes” on page 68

About this task

Instructions and sample files for deploying content services on Kubernetes are provided in the following GitHub repository https://github.com/ibm-ecm/container-samples/tree/5.5.3.

Completing post-deployment startup tasks for an upgrade

After you run the container deployment, you perform additional tasks to update and start your P8 domain.

Completing post-deployment steps for a Content Platform Engine container upgrade

After you upgrade or change your Content Platform Engine environment to a container deployment, you must complete additional configuration steps before you can use the new environment.

Before you begin

The topic Chapter 4, “Preparing the environment to upgrade an existing FileNet P8 domain,” on page 99 includes a subset of steps for preparing your existing file servers for use with containers. These steps must be completed before you complete these steps.

About this task

Ensure the CPE service has completed the deployment by performing the remaining verification procedures in the following topic: Verifying the Content Platform Engine deployment. Then complete the configuration by using the following steps.

Procedure

1. If SSL is enabled in the existing system for the Content Platform Engine to Content Search Services communications, use your own SSL keystore and certificate files in place of the default files that are provided with the Content Platform Engine container deployment.

   Add the files `keystore.jks` and `trustore.jks` from your existing system to the configuration overrides directory that you created on the Content Platform Engine configuration volume, for example: `/cpecfgstore/cpe/configDropins/overrides`.

   For more details, see step 2 in the following topic: “Enabling custom SSL certificates for Content Search Services” on page 110
2. For any standard file store areas you have defined, edit the root path property to reflect the mountPath as viewed from inside the container.
   a) Log in to the Administration Console for Content Platform Engine, and navigate to **Domain** > **Object stores** > ***object_store*** > **Administrative** > **Storage** > **Storage Areas**.
   b) Choose the File Storage Area.
   c) On the **Properties** tab, edit the property **`Root Directory path`** based on the Content Platform Engine filestore volume mapping to the mountPath `/opt/ibm/asa`.

      For example, if a file storage area exists at `/data/FMdomain_Marketing/FS1` and the Content Platform Engine filestore volume claim is to `/data/FMdomain_Marketing`, the new value for the Root Directory Path will be `opt/ibm/asa/FS1`.

   d) Repeat these steps for every file storage area in the FileNet P8 domain.

3. For any advanced storage device that is defined as a file system storage device, edit the advanced storage device root path.

   a) Log in to the Administration Console for Content Platform Engine, and navigate to **Domain** > **Object stores** > ***object_store*** > **Administrative** > **Advanced Storage** > **Advanced Storage Devices**.

   b) Choose the Advanced Storage Device.

   c) On the **Configuration** tab, edit the value for **`Root Directory path`** based on the Content Platform Engine filestore volume mapping to the mountPath `/opt/ibm/asa`.

      For example, if an advanced storage device exists at `/data/FMdomain_Marketing/ASAD1` and the Content Platform Engine filestore volume claim is to `/data/FMdomain_Marketing`, the new value for the Root Directory Path will be `/opt/ibm/asa/ASAD1`.

   d) Repeat these steps for every advanced storage device in the FileNet P8 domain.

Completing post-deployment steps for a Content Search Services container upgrade

After you upgrade or change your Content Search Services environment to a container deployment, you must complete additional configuration steps before you can use the new environment.

Before you begin

Review the following information to establish useful context for the steps you perform to configure your new container-deployed Content Search Services server: Configuring IBM Content Search Services.

The topic Chapter 4, "Preparing the environment to upgrade an existing FileNet P8 domain," on page 99 includes a subset of steps for preparing your existing file servers for use with containers. These steps must be completed before you complete these steps.

Procedure

To configure Content Search Services after an upgrade to containers:

1. For each existing index area, edit the index areas root path values to reflect the mountPath for the Content Search Services containers:

   a) Log in to the Administration Console for Content Platform Engine using the credentials of the object store owner.

      If you use different credentials, you might not have the required permissions for the object store.

   b) In the Administration Console for Content Platform Engine, navigate to **Domain** > **Object stores** > **object_store** > **Administrative** > **Index Areas**.

   c) Choose an index area to edit.

   d) On the **General** tab, edit the property **Root Directory** based on the Content Search Service index area volume mapping to the `mountPath/opt/ibm/indexareas`.

      For example, if an index area **`IndexArea1`** exists at `/data/Marketing_Indexes/IndexArea1` and the Content Search Services index store volume claim is to `/data/Marketing_Indexes`, the new value for the Root Directory is `/opt/ibm/indexareas/IndexArea1`.

   e) Repeat these steps for every index area in the FileNet P8 domain.

2. Change the ownership of the existing index areas so that they can be accessed by the Content Search Services services that are deployed as containers.

   Modify the index areas parent directory and all children to have ownership of 50001: 50000 (for V5.5.2) or 501:500 (for V5.5.1).

   (V5.5.2) Use a command like the following example, where the sample parent directory is `/data/Marketing_Indexes`:

   ```
   chown –Rf 50001:50000 /data/Marketing_Indexes
   ```

(V5.5.1) Use a command like the following example, where the sample parent directory is `/data/Marketing_Indexes`:

```
chown –Rf 501:500 /data/Marketing_Indexes
```

If a single parent directory for all index areas does not exist, create the parent directory in the file system and reference from it to the pre-existing data. As an alternative, you can also move the index areas so that they are all located under a single parent directory.

3. If your container deployment exists alongside an on-premises deployment, modify the file system that holds the index areas to provide a reference for the Content Search Service services that are deployed as Java applications.

   The reference enables the services to see the index area parent directory as `/opt/ibm/indexareas`.
4. Enable SSL between the text search servers and Content Platform Engine.

   In all the following cases, the files holding the certificates are added to the volume that you created for the Content Search Services configuration volume, for example, `/contentServices/CSS/css-data`.

   a) If the existing system enabled SSL for the Content Platform Engine to Content Search Services communications, you must use your own SSL keystore and certificate files in place of the default file that is provided with the Content Search Services container deployment. Add the `fileskeystore.jks` and `truststore.jks` from your existing system to the CSS configuration volume.

   b) If the existing system did not previously enable SSL for the Content Platform Engine to Content Search Services communication, the sample `cssSelfsignedServerStore` file provided by default with the Content Search Services container can be downloaded and used. Add the downloaded `cssSelfsignedServerStore` file to the Content Search Services configuration volume.

   c) Alternatively, you can generate and store your own certificates in the Content Search Services configuration volume as described in the topic Configuring SSL for IBM Content Search Services
5. Modify the existing FileNet P8 domain configuration to incorporate the newly deployed Content Search Service services:

   a) Obtain the configuration details for the new CSS services in the container deployment:

      1) In IBM Cloud Private, check the deployment page to determine the pod name for your CSS server deployment.
      2) From the command line, use your CSS pod name and run the following command:

         ```
         kubectl exec -it pod pod_name bash
         ```

      3) From the command line, change to the `/opt/IBM/ContentSearchServices/CSS_Server/bin` directory, and run the following command to get the authentication token:

         ```
         ./configTool.sh printToken -configPath https://www.ibm.com/support/knowledgecenter/en/
         SSNW2F_5.5.0/config
         ```

         Make a note of the token so you can use it during the configuration.
      4) Exit out of the pod.
      5) From the command line, run the following command to get information about the pod:

         ```
         kubectl describe pod pod_name
         ```

         Make a note of the IP and port (8199) so you can use it during the configuration.

   b) Configure a text search server for each IBM Content Search Services deployment by navigating to **Global Configuration** > **Administration** > **Text Search Servers** > **New Text Search Server**, and complete the wizard:

      Server state:

- Ensure all Content Search Services servers in the containerized environment are configured for `Dual: IndexAndSearch` mode.

  Server parameters:

- Set the **Host name** and **Port** values to the IBM Cloud Private pod IP and the cssssl port as seen from IBM Cloud Private. By default the cssssl port value is 8199.
- Add the authentication token.

  This encryption key is used to encrypt the password during text index backup and restore operations.

  Communication security:

- Check **Enable use of the Secure Socket Layer (SSL) protocol**.
- Uncheck **Validate the SSL server certificate** and **Validate the SSL certificate host**.

  Server association:

- Set the **Affinity Group** value to the same affinity group that is set for your existing on-premises CSS server.

c) Save the server settings.

d) Repeat the preceding steps for all the newly deployed Content Search Service services.

6. Expand **Index Areas**, and update the path to point to the container's index area.

   To find the path, run the following command:

   ```
   kubectl exec -it pod_name bash
   ```

   Examine the output and look for the path value for index areas, for example, `/opt/ibm/indexareas`.
7. Alter any affinity groups that might exist to incorporate the additional text search servers.

   Navigate to **Domain** > **Global Configuration** > **Administration** > **Text Search Servers**, and choose a text search server to edit.

   - For a move to container-only deployment:

     a. Ensure that one or more text search servers created for the containerized Content Search Services deployments are assigned as members in existing affinity groups.

     b. Remove the old text search servers from any affinity groups they might belong to.

     c. Leave the old text search servers disabled and schedule them for future deletion.
   - For containers alongside an on-premises installation:

     a. Verify that the existing text search servers are already upgraded to be the same version as the version that is used by the containers.

     b. Restart the existing text search servers.

     c. Assign text search servers as needed to rebalance affinity groups as needed using both Content Search Service services that are deployed as Java applications and Content Search Service services that are deployed using containers.
8. Re-enable text indexing for the object stores.

   For more information, see the following: Starting the IBM Content Search Services index dispatcher.
9. Verify the Content Search Services deployment for an existing FileNet P8 domain.

   For more information, see the following: Verifying the IBM Content Search Services upgrade

Completing post-deployment steps for an IBM Content Navigator upgrade

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

About this task

After you update your IBM Content Navigator to a container deployment, use the IBM Content Navigator administration desktop 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 in the on-premises system before upgrade is `*install_dir*/ibm/viewerconfig/logs/daeja.log`, where *install_dir* is the directory where IBM Content Navigator is installed. Update the path 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 desktop, click **Sync Services**.
   b) Update the default value for the public service URL.

      The URL is based on the URL for IBM Content Navigator, and includes the server IP address and port for the IBM Cloud Private environment (`http://ICP_IP_Address:30557/navigator`). For the public service URL, change the `/navigator` to `/sync/notify`, for example: `http://ICP_IP_Address:30557/sync/notify`.

      Note: If you are using container versions of the applications side-by-side with on-premises versions, you have the option to leave your Sync Services URL to the on-premises setting, or to change it to the public service URL.
4. For any IBM Content Navigator instance that references a FileNet P8 repository that is managed by a container deployment of Content Platform Engine, modify the server URL value to incorporate the CPE service:
   a) Check the login URL for the Administration Console for Content Platform Engine in the container deployment, for example, `http://9.30.118.223:30954/acce`.
   b) Log into the IBM Content Navigator administration desktop.
   c) Open the **Repositories** view, select the FileNet P8 repository, and choose an action of edit.
   d) Modify the Server URL field to reflect the host and port values from the Administration Console for Content Platform Engine, for example, `http://9.30.118.223:30954/wsi/FNCEWS40MTOM`.
   e) Repeat this procedure for every FileNet P8 repository in the IBM Content Navigator configuration.
5. Update custom plug-ins:
   a) Move the IBM Content Navigator plug-in files from your on-premises location to the following path in the IBM Content Navigator volume: `/opt/ibm/plugins`.
   b) Open the administration desktop in the web client.
   c) Click **Plug-ins**, choose the plug-in you want to update, and then click **Edit**.
   d) Reload the plug-in from the new path in the IBM Content Navigator volume.

      The plug-in file must be in the `/opt/ibm/plugins` path in the volume.
   e) Provide any additional configuration updates that the plug-in requires.
   f) Save your changes.

Chapter 7. (V5.5.2 or later) Upgrading container deployments

Starting in V5.5.2, you can upgrade your container deployments to the latest version by deploying new versions of the container images in IBM Cloud Private.

Before you begin
Make a backup of your content services configuration volumes and folders, as well as your FileNet P8 domain and data. For details, see the following information: Backing up the data in your FileNet P8 domain

About this task

(V5.5.2) An upgrade to V5.5.2 is similar in process to a new installation. You download the containers, create an archive image, load the image into IBM Cloud Private, and deploy the containers by using the Business Automation Configuration Container.

(V5.5.3) An upgrade to V5.5.3 uses Helm charts to upgrade to the latest container images.

Depending on the upgrade choices you make for IBM Cloud Private, you might also need to take preparation steps for the upgrade that are similar to some of the preparation steps for a new install. For example, you might need to re-create the persistent volumes and persistent volume claims.

(V5.5.2) In both scenarios, you must upgrade to IBM Cloud Private V3.1.0 before you start the container upgrade.

(V5.5.3) You must upgrade to IBM Cloud Private V3.1.2 before you start the container upgrade.

Check the IBM Software Product Compatibility Report for the appropriate versions of supporting software.

While you are upgrading your container environment, you must restrict all user access to the content services. Be sure to plan the upgrade for a time when it will be most convenient for your users.

Migrating to V5.5.4 with operators

Because of a change in container deployment method, an upgrade to V5.5.4 is not available. Instead, you deploy the new version and designate your existing data and configuration settings.

About this task
If you want to move to V5.5.4 from an earlier container version, you follow the steps for a new deployment using the new operator Framework. For more information, see the readmes for Migrating in the Git Hub location:

Table 19. Options for deployment with the operator

Platform	Where to go
Managed Red Hat OpenShift on IBM Cloud Public	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/roks/README.md
Red Hat OpenShift	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/ocp/README.md
Certified Kubernetes	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/k8s/README.md

(V5.5.3 and earlier) Creating the content services archive file for upgrade on IBM Cloud Private

To make the new images for content services containers and the updated IBM Business Automation Configuration Container available for deployment in IBM Cloud Private, you must first create an archive file of the images. You then load the created archive file into IBM Cloud Private.

Before you begin

When running in some PAAS environments, you might need to configure the NO_PROXY setting in advance. Otherwise, you might encounter an error while loading the chart archive.

Run the following command:

```
# export NO_PROXY=localhost,127.0.0.1,domain_suffix,ICP_CLUSTER_NAME,
SC_K8S_CLUSTER_NAME
```

For example:

```
# export NO_PROXY=localhost,127.0.0.1,.suffix.ibm.com,mycluster,mycluster.icp
```

About this task

(V5.5.3) See the V5.5.3 download doc to find the part numbers for the components that you need for this task, as well as the link to Passport Advantage: http://www.ibm.com/support/docview.wss?uid=ibm10881658

An upgrade to V5.5.3 uses Helm charts to deploy the new container images. You can load the archive before you are ready to upgrade the content services containers, but after you have upgraded to IBM Cloud Private 3.1.2.

(V5.5.2) See the V5.5.2 download doc to find the part numbers for the components that you need for this task, as well as the link to Passport Advantage: https://www.ibm.com/support/docview.wss?uid=ibm10741447

An upgrade to V5.5.2 containers resembles a new installation, so you must create and load the archive of updated container images before you can run the deployment. You can load the archive before you are ready to upgrade the content services container, but after you have upgraded or installed IBM Cloud Private 3.1.0.

Procedure

To create the container image archive file:

1. From Passport Advantage, download the IBM Business Automation Configuration Container compressed file, *ibacc-part-number*.tar.gz, and use the tar command to extract the file to a folder that you specify:

   ```
   tar -zxvf ibacc-part-number.tar.gz -C <output_folder>
   ```

   (V5.5.3) The following example shows the output folder contents:

   ```
   <output_folder>
   ├── charts
   │   └── ibm-dba-content-prod-3.0.0.tgz
   ├── images
   │   ├── a3ea3c3ab12581c6fa853775a4f466db7a91704fa0f7bf5daa04f5badd8c668e.tar.gz
   │   ├── 56587cfdf24ae5bba72efc73d24bd3788ca20a676c997798447c7c03fce330f1.tar.gz
   │   ├── fdba4cd1b6287593f7d3e186d2c013f92d6a7d927b6eb134407b0bc0ec51c2fe.tar.gz
   │   └── 2e88e40f634f4bf82ceb98056759a45356a82116bc695ec2f2ee47d24d0a0119.tar.gz
   ├── manifest.json
   └── manifest.yaml
   ```

   (V5.5.2) The following example shows the output folder contents:

```
<output_folder>
├── charts
│   └── ibm-dba-content-prod-2.0.0.tgz
├── images
│   ├── 6bc9b8634a9af27b45d3315d8a4373979aaa492244c6590c2754df1b40df3858.tar.gz
│   ├── 78395a35e0e26ba1868c925fcf31ca7965f52a2008d8e1447479a4ef18075938.tar.gz
│   ├── c5d8397915ffc3a9993b8e621f99d704ae944d0fdf5a9f347e368b3c06c7e8a5.tar.gz
│   └── f86d4686c177a6d817b3edc0690bb3f9bfd2c9cbe48909172a937a243f7ea802.tar.gz
├── manifest.json
└── manifest.yaml
```

The long file names are the SHA256 IDs of the specified Docker images.

2. From Passport Advantage, download the Docker images for the content services component containers:
   - *CPE-container-part-number*.tar
   - *ICN-container-part-number*.tar
   - *CSS-container-part-number*.tar
3. From Fix Central, download the Docker image for Content Management Interoperability Services.
   - (V5.5.3)*3.0.4.0-CMIS-Container-IF007*.tar
   - (V5.5.2)*3.0.4.0-CMIS-Container-IF003*.tar
4. Run the tar command to extract the original content services image archive tar.gz files:

   ```
   # tar xvf CPE-container-part-number.tar
   ```

   Repeat the command for each of the remaining three downloaded content service container image tar files:
   - *ICN-container-part-number*.tar
   - *CSS-container-part-number*.tar
   - (V5.5.3)*3.0.4.0-CMIS-Container-IF007*.tar
   - (V5.5.2)*3.0.4.0-CMIS-Container-IF003*.tar

   The command results in a tar.gz file from each container TAR file, and two tar.gz files from the IBM Content Navigator container TAR file. Note that the tar.gz files can have long IDs.
5. Move the content services image archive files (*.tar.gz) to the `images` folder in the output folder that you created when you extracted the IBM Business Automation Configuration Container tar file:

   For example, the following command moves the Content Platform Engine image:

   ```
   mv <cpe_image_tar>.gz  <output_folder>/images
   ```

   Repeat the command for each of the remaining archive image files that you extracted. Remember that the IBM Content Navigator TAR file extraction results in two *image*.tar.gz files that must both be moved to the *output_folder*/images folder.

   (V5.5.3) The contents of the output folder now looks like the following example:

   ```
   <output_folder>
       ├── charts
       │   └── ibm-dba-content-prod-3.0.0.tgz
       ├── images
       │   ├── a3ea3c3ab12581c6fa853775a4f466db7a91704fa0f7bf5daa04f5badd8c668e.tar.gz
       │   ├── 56587cfdf24ae5bba72efc73d24bd3788ca20a676c997798447c7c03fce330f1.tar.gz
       │   ├── fdba4cd1b6287593f7d3e186d2c013f92d6a7d927b6eb134407b0bc0ec51c2fe.tar.gz
       │   ├── 2e88e40f634f4bf82ceb98056759a45356a82116bc695ec2f2ee47d24d0a0119.tar.gz
       │   ├── 1ba75e4174939b96b17335a3dab336a96e6fcc3df75d8ac0fcca6d7fdd4c0eb1.tar.gz
       │   ├── 395b70bc1a9944a4d01ac61eb9e78db1be76fad1e067f2110b883a2f684dc4fb.tar.gz
       │   ├── 194262f8351e6349f8d0058cdd00fa8f311fc0962a7ae7e193be248d0b606cea.tar.gz
       │   ├── 0e33b27dd1138f23ce825019e02fe3810091685b8e9de5dce04ae8ee17ed27ae.tar.gz
       │   └── 004bdd4f629a271ec551aeede1c43295ec1cfd766de57fd6d7083c5e35863556.tar.gz
       ├── manifest.json
       └── manifest.yaml
   ```

   (V5.5.2) The contents of the output folder now looks like the following example:

```
<output_folder>
    ├── charts
    │   └── ibm-dba-content-prod-2.0.0.tgz
    ├── images
    │   ├── 5529941196f0ec9327ff5056ab723643a3c094895320b2179a47876e51a15302.tar.gz
    │   ├── 5d59dfb451f7c347660d792c97625b108fe243bccee0498e5bfa862d4e01e4c3.tar.gz
    │   ├── 6bc9b8634a9af27b45d3315d8a4373979aaa492244c6590c2754df1b40df3858.tar.gz
    │   ├── 78395a35e0e26ba1868c925fcf31ca7965f52a2008d8e1447479a4ef18075938.tar.gz
    │   ├── 8fefb76b3abd78f383c35eb49630675925ccfcf6f17d50bf112528b4f57e445c.tar.gz
    │   ├── b343d3e7d3ad9d8d19116376997eefe2569db2f2aa958e279aa3844ea8f9c602.tar.gz
    │   ├── c5d8397915ffc3a9993b8e621f99d704ae944d0fdf5a9f347e368b3c06c7e8a5.tar.gz
    │   ├── d2cba67ac4c1f5f5b9ea14eb2ab010f8fadc625b828e6b8f1d2e1f00e6e0c3f1.tar.gz
    │   └── f86d4686c177a6d817b3edc0690bb3f9bfd2c9cbe48909172a937a243f7ea802.tar.gz
    ├── manifest.json
    └── manifest.yaml
```

6. Change to the *output_folder* location:

```
# cd output_folder
```

7. Run the tar command to compress the *output_folder*:

```
# tar -zcvf icp_charts_images.tgz output_folder/*
```

   Important: Ensure that the content and folder and file structure inside the archive file are the same as what is in the *output_folder* location.

8. Use the command line to log in to IBM Cloud Platform:

   For V5.5.3, use the following command:

```
cloudctl login -a https://<icp_ip_address>:8443 --skip-ssl-validation
```

   For V5.5.2, use the following command:

```
cloudctl login -a https://<icp_ip_address>:8443 --skip-ssl-validation
```

9. Log in to your cluster:

```
docker login mycluster.icp:8500
```

10. Load the archive file that you created into IBM Cloud Private:

   For V5.5.3, use the following command:

```
cloudctl catalog load-archive --archive icp_charts_images.tgz --registry mycluster.icp:8500
```

   For V5.5.2, use the following command:

```
cloudctl catalog load-archive --archive icp_charts_images.tgz
--clustername mycluster.icp --namespace ecm-namespace

or

cloudctl catalog load-archive --archive icp_charts_images.tgz
--registry mycluster.icp:8500 --namespace your_namespace
```

   In the command, replace any variables for cluster, namespace, and so on with actual values from your environment.

Upgrading to V5.5.3 on IBM Cloud Private

To upgrade your container deployments to V5.5.3, you must upgrade the Helm releases with the new 5.5.3 container images.

Before you begin

1. Plan your upgrade for a time when it is convenient to restrict all user access to content services.

2. If you have not already done so, make a backup of your environment folders and volumes on IBM Cloud Private.
3. Download the new IBM Passport Advantage (PPA) archive and load the container images to a registry in IBM Cloud Private. After you load the images, you can view them in IBM Cloud Private.
4. Create a new persistent volume and persistent volume claim for the Content Search Services custom store, /css/customstore.

   Set the folder ownership to 50001:50000:

   ```
   chown –Rf 50001:50000 /css/customstore
   ```

   The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the Content Search Services custom store.

   Persistent volume:

   ```
   apiVersion: v1
   {
     "kind": "PersistentVolume",
     "apiVersion": "v1",
     "metadata": {
       "name": "css-icp-customstore-pv",
       "labels": {}
     },
     "spec": {
       "storageClassName": "css-icp-customstore",
       "capacity": {
         "storage": "1Gi"
       },
       "accessModes": [
         "ReadWriteMany"
       ],
       "persistentVolumeReclaimPolicy": "Retain",
       "nfs": {
         "path": "/mnt/nfsshare/css/customstore",
         "server": "<NFS_SERVER>"
       }
     }
   }
   ```

   Persistent volume claim:

   ```
   apiVersion: v1
   {
     "kind": "PersistentVolumeClaim",
     "apiVersion": "v1",
     "metadata": {
       "name": "css-icp-customstore",
       "namespace": "default"
     },
     "spec": {
       "storageClassName": "css-icp-customstore",
       "resources": {
         "requests": {
           "storage": "1Gi"
         }
       },
       "accessModes": [
         "ReadWriteMany"
       ]
     }
   }
   ```

5. (Upgrades from V5.5.2) Update the permissions on the Content Management Interoperability Services folders:

   For each of the folders, set the ownership to 50001:50000:

   ```
   chown –Rf 50001:50000 /cmiscfgstore
   ```

6. (Upgrades fromV5.5.1) If you are deploying the IBM Content Navigator container, create the persistent volume and persistent volume claim for Aspera.

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-icp-asperastore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /home/cfgstore/icn/aspera
    server: <NFS_SERVER>
  persistentVolumeReclaimPolicy: Retain
  storageClassName: icn-icp-asperastore-pv
```

Persistent volume claim:

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

7. (Upgrades from V5.5.1) Update the permissions on all of the persistent volume folders in your environment:

   For each of the folders, set the ownership to 50001:50000:

   ```
   chown –Rf 50001:50000 /cpecfgstore
   ```

About this task

Each release on IBM Cloud Private is created from a specific Helm chart. To upgrade a release that is created with a previous chart version, you extract the new helm charts and then upgrade the releases with the new container images.

The name of the container image to use to get the helm charts is listed here, along with the image tag.

- Configuration container job container: `ibacc-icp:3.0.0`

The names and tags of the new container images are listed here.

- Content Platform Engine container image: `cpe:ibm-dba-contentservices-3.0.0`
- IBM Content Navigator container image: `icn:ibm-dba-navigator-3.0.0`
- Content Search Services container image: `css:ibm-dba-contentsearch-3.0.0`
- IBM Content Management Interoperability Services container image: `cmis:ibm-dba-cscmis-1.7.0`

During the process of upgrading your container images, you must also perform the standard checks and post upgrade configuration tasks to ensure that your content services environment continues to function as it did before the upgrade.

Be sure to plan the timing of the upgrade to ensure minimal disruption to your users.

If you are using IBM Content Navigator with Content Platform Engine, coordinate the timing of your upgrade for that container to coincide with your content services upgrades. When you change permissions on the persistent volumes for the Content Platform Engine, the IBM Content Navigator application no longer has access to the Content Platform Engine. You can change the permissions on the persistent volumes at the same time for your IBM Content Navigator container, and then complete the upgrade steps for that container after you complete this upgrade procedure.

The following steps assume that you have all four content services containers in your environment. If you do not use all of the containers, skip over the steps for the containers that do not apply.

Procedure

1. Extract the Helm charts from the PPA package.

   a. Prepare a folder for the archive.

   ```
   mkdir -p /helm-charts \
   chown -R 50001:50000 /helm-charts \
   cd /helm-charts
   ```

   b. Run a docker command like the following example to get the Helm charts from the PPA package.

   ```
   docker run -v "$(pwd)":/helm-charts mycluster.icp:8500/namespace/image_name:tag
   cp /opt/ibm/icp/ecm-helm-charts/CONTENT/archive_name.tgz /helm-charts
   ```

   The names of the Helm charts are listed here.

   - `ibm-dba-contentservices-3.0.0.tgz`
   - `icn:ibm-dba-navigator-3.0.0`
   - `ibm-dba-contentsearch-3.0.0.tgz`
   - `ibm-dba-cscmis-1.7.0.tgz`

   Run the following commands get all of the Helm charts:

   ```
   docker run -v "$(pwd)":/helm-charts mycluster.icp:8500/default/ibacc-icp:3.0.0
   cp /opt/ibm/icp/ecm-helm-charts/CONTENT/ibm-dba-contentservices-3.0.0.tgz /helm-charts

   docker run -v "$(pwd)":/helm-charts mycluster.icp:8500/default/ibacc-icp:3.0.0
   cp /opt/ibm/icp/ecm-helm-charts/CONTENT/ibm-dba-navigator-3.0.0.tgz /helm-charts

   docker run -v "$(pwd)":/helm-charts mycluster.icp:8500/default/ibacc-icp:3.0.0
   cp /opt/ibm/icp/ecm-helm-charts/CONTENT/ibm-dba-contentsearch-3.0.0.tgz /helm-charts

   docker run -v "$(pwd)":/helm-charts mycluster.icp:8500/default/ibacc-icp:3.0.0
   cp /opt/ibm/icp/ecm-helm-charts/CONTENT/ibm-dba-cscmis-1.7.0.tgz /helm-charts
   ```

2. Ensure that access to your content services system is restricted or stopped.
3. Run the `helm upgrade` command for the Content Platform Engine container.

   The following command upgrades a release with the name `cpe` with the new Content Platform Engine container image.

   ```
   helm upgrade cpe /helm-charts/ibm-dba-contentservices-3.0.0.tgz --tls --reuse-values --set
   image.repository=mycluster.icp:8500/default/cpe,image.tag=ga-553-
   p8cpe,resources.requests.cpu=500m,resources.requests.memory=512Mi,resources.limits.cpu=1,reso
   urces.limits.memory=1024Mi,imagePullSecrets.name=admin.registrykey
   ```

   Note: The requirements for the additional command parameters might be different for your environment. Edit the values appropriately before you run the command.

   Wait for the upgrade to complete. You can check the **Deployments** page in IBM Cloud Private to view the status of the upgrade.

4. When the deployment is completed, check the status of the Content Platform Engine:

   a) Determine the server IP and port for the Content Platform Engine container deployment.

In the IBM Cloud Private console, browse to **Workloads** > **Deployments**, and locate the Content Platform Engine deployment, for example `cpe-ibm-contentservices`. Click **Launch HTTP** to open the URL that displays the server IP and port number.

Use these values for the subsequent status checking steps.

b) Check the Content Platform Engine Startup Context (Ping Page).

Browse to `http://`*`server`*`:`*`port`*`/FileNet/Engine`, where:

- *server* is the host name of the machine where Content Platform Engine is deployed.
- *port* is the HTTP port where Content Platform Engine is deployed.

c) Check the automatic upgrade status for Content Platform Engine data.

1) Enter the URL for the automatic upgrade status page into your browser `http://`*`server_name`*`:`*`port`*`/FileNet/AutomaticUpgradeStatus`.
2) Review the status for the FileNet P8 domain and object stores.

5. Run the `helm upgrade` command for the IBM Content Navigator container.

The following command upgrades a release with the name `icn` with the new IBM Content Navigator container image.

```
helm upgrade icn /helm-charts/ibm-dba-navigator-3.0.0.tgz --tls --reuse-values --set
image.repository=mycluster.icp:8500/default/css,image.tag=ga-553-
p8css,resources.requests.cpu=500m,resources.requests.memory=512Mi,resources.limits.cpu=1,reso
urces.limits.memory=1024Mi,imagePullSecrets.name=admin.registrykey
```

Note: The requirements for the additional command parameters might be different for your environment. Edit the values appropriately before you run the command.

Wait for the upgrade to complete. You can check the **Deployments** page in IBM Cloud Private to view the status of the upgrade.

6. Check status of IBM Content Navigator:

Verify that you can access IBM Content Navigator at the following URL: `http://hostname:icn_port/navigator`

Verify that the Content Platform Engine server information is correct.

a. Log in to the Navigator Admin desktop: `http://<icp_ip_address>:<navigator_port>/navigator/?desktop=admin`
b. Navigate to **Repositories**.
c. Select the repository, and click **Test Connection** to verify the communication with the Content Platform Engine services are functional.

If the test fails, click **Edit** and change the **Server URL** to update the new *icp_ip_address* and *CPE_port* values:

```
icp_ip_address:CPE_port\\
```

7. Run the `helm upgrade` command for the Content Search Services container.

The following command upgrades a release with the name `css` with the new Content Search Services container image.

```
helm upgrade dbamc-css /helm-charts/ibm-dba-contentsearch-3.0.0.tgz  --reuse-values --set
image.repository=<image_repository_url>:5000/dbamc/css,image.tag=ga-553-
p8css,imagePullSecrets.name=admin.registrykey,resources.requests.cpu=500m,resources.requests.
memory=512Mi,resources.limits.cpu=8,resources.limits.memory=8192Mi,log.format=json,dataVolume
.nameforCSSCustomstore=custom-stor,dataVolume.existingPVCforCSSCustomstore=css-icp-
customstore
```

Note: The requirements for the additional command parameters might be different for your environment. Edit the values appropriately before you run the command.

Wait for the upgrade to complete. You can check the **Deployments** page in IBM Cloud Private to view the status of the upgrade.

8. Verify your Content Search Services upgrade by using the following instructions Verifying the IBM Content Search Services upgrade.
9. Run the `helm upgrade` command for the Content Management Interoperability Services container.

   The following command upgrades a release with the name `cmis` with the new Content Management Interoperability Services container image.

   ```
   helm upgrade cmis /helm-charts/ibm-dba-cscmis-1.7.0.tgz --tls --reuse-values --set
   image.repository=mycluster.icp:8500/default/cmis,image.tag=ga-305-
   cmis,resources.requests.cpu=500m,resources.requests.memory=512Mi,resources.limits.cpu=1,resou
   rces.limits.memory=1024Mi,imagePullSecrets.name=admin.registrykey
   ```

   Note: The requirements for the additional command parameters might be different for your environment. Edit the values appropriately before you run the command.

   You can check the **Deployments** page in IBM Cloud Private to view the status of the upgrade.

Upgrading to V5.5.2 on IBM Cloud Private

You can upgrade on a new IBM Cloud Private server where you have installed the new version of IBM Cloud Private software.

Before you begin

- Make a backup of your content services configuration volumes and folders, as well as your FileNet P8 domain and data.
  - Configuration volumes and folders
  - FileNet P8 domain and data: Backing up the data in your FileNet P8 domain
- Make sure that you have downloaded the container images and created the archive file as described in "(V5.5.3 and earlier) Creating the content services archive file for upgrade on IBM Cloud Private" on page 130. You upload this archive image after you upgrade IBM Cloud Private.

About this task

As part of this process, before you upgrade the content services containers, you upgrade to IBM Cloud Private V3.1.0. During the upgrade of IBM Cloud Private and your content services containers, your content services applications are unavailable for use. As a result, when you are upgrading your container environment, you must restrict all user access to the content services. Be sure to plan the upgrade for a time when it will be most convenient for your users.

These steps assume that you are installing the new version of IBM Cloud Private on a new server. If you are upgrading in place, the steps might vary. See the IBM Cloud Private upgrade procedures for more information.

Check the IBM Software Product Compatibility Report for the appropriate versions of supporting software.

Important: Do not enable the Initialization and Verification options in the Business Automation Configuration Container user interface. These services do not apply in the upgrade case.

Procedure

To upgrade the content services containers on a new IBM Cloud Private server:

1. On the new server, install IBM Cloud Private V3.1.0. See Installing IBM® Cloud Private 3.1.0 for details.
2. On your previous IBM Cloud Private environment, use the following commands to obtain all the details about your PVs, PVCs, and configuration information:

   (To determine your Helm release names, in the IBM Cloud Private console menu, go to **Workloads** > **Helm Releases**, locate your container, and use the value in the **NAME** column.)

For Content Platform Engine:

```
bx pr login -a https://host:8443 --skip-ssl-validation
       helm get <cpe-helm-release> values --tls > cpeoutput.txt
```

For Content Search Services:

```
bx pr login -a https://host:8443 --skip-ssl-validation
       helm get <css-helm-release> values --tls > cssoutput.txt
```

For Content Management Interoperability Services:

```
bx pr login -a https://host:8443 --skip-ssl-validation
       helm get <cmis-helm-release> values --tls > cmisoutput.txt
```

For IBM Content Navigator:

```
bx pr login -a https://host:8443 --skip-ssl-validation
       helm get <icn-helm-release> values --tls > icnoutput.txt
```

3. Use the information in the `output.txt` files to help you as you re-create all of the persistent volumes and persistent volume claims on the new server.

   For a reminder of what PVs and PVCs are required, see "Creating volumes and folders for deployment on IBM Cloud Private" on page 48.
4. Create a new Persistent Volume and Persistent Volume Claim pair for Aspera:

   Table 20. Volumes, volume claim, and folder for Aspera for IBM Content Navigator

Volume purpose	**Example Folder to Create**	**Example Volume and Volume Claim to Create**	**mountPath as seen by container**
(V5.5.2) IBM Content Navigator storage for Aspera	/icnasperastore	icn-icp-asperastore	`/opt/ibm/aspera`

   Set the ownership to 50001:50000:

   ```
   chown –Rf 50001:50000 /cpecfgstore
   ```
5. In your storage environment, update the permissions for all of your existing content services volumes (with the exception of CMIS) from 501:500 to 50001:50000.

   For example, the following command sets the permissions for the default Content Platform Engine folder.

   ```
   chown -R 50001:50000 /cpecfgstore
   ```

   Important: This permission change does not apply for the volumes that are used by Content Management Interoperability Services. The permissions for these volumes remain set to 501:500.

   Make this change for all of the volumes for your content services containers except for the CMIS PVCs. See the following for details about the relevant volumes to update: Creating volumes and folders for deployment.
6. Override the permissions for all of the Content Management Interoperability Services volumes with write permission for all users.

   For example, the following command runs the `chmod` command on a CMIS volume that is named `cmiscfgstore`:

   ```
   chmod -R 777 /cmiscfgstore
   ```
7. Verify that you have created the content services archive file with the new images to your new IBM Cloud Private server.

For details, see "(V5.5.3 and earlier) Creating the content services archive file for upgrade on IBM Cloud Private" on page 130.

8. Use the command line to log in to IBM Cloud Private on the new V3.1.0 server:

   ```
   cloudctl login -a https://<icp_ip_address>:8443 --skip-ssl-validation
   ```

9. Log in to your cluster:

   ```
   docker login mycluster.icp:8500
   ```

10. Load the archive file that you created into IBM Cloud Private:

    ```
    cloudctl catalog load-archive --archive icp_charts_images.tgz
    --clustername mycluster.icp --namespace ecm-namespace
    ```

    In the command, replace any variables for cluster, namespace, and so on with actual values from your environment.
11. Deploy the Business Automation Configuration Container tool.

    Use the steps in "Deploying the IBM Business Automation Configuration Container" on page 67 to complete this process.
12. On the previous IBM Cloud Private server, stop your containers in the following order:
    a. Stop IBM Content Navigator.
    b. Stop Content Management Interoperability Services.
    c. Stop Content Search Services.
    d. Stop Content Platform Engine.
13. On the new IBM Cloud Private (V3.1.0) server, access the configuration tool by clicking **Workloads** > **Helm release** > ***release_name*** > **Services**.

    The **Node port** value provides a link that you can click to start the configuration tool.
14. Start the deployment, provide a name, and select IBM Cloud Private as the Cluster type.
15. Provide the configuration details about your IBM Cloud Private environment.
16. Choose only the Content Platform Engine component to deploy.

    Enter the configuration details from your existing configuration that are requested by the tool. If you used logging and monitoring, add those to the configuration. Do not enable Initialization and Verification.
17. Enter the configuration information for the previously configured database and directory server that you use for your container environment.

    You must also upload the required JDBC driver.
18. After you enter and review all the required configuration values, click **Deploy**.

    The deployment results page displays. The page contains three parts:

    - The progress of the deployment, whether finished or in progress. Once started, the **Action** link is enabled and you can click the link to download the detailed log file. There is also a re-run option if the task fails.
    - The service URL. After the provision task finishes, the available service URL displays
    - Components logs. You can check the real time log results.

    You can use the download icon to download the configuration files.
19. When the deployment is completed, check the status of the Content Platform Engine:

    a) Determine the server IP and port for the Content Platform Engine container deployment.

       From the IBM Cloud Private V3.1.0 console, browse to **Workloads** > **Deployments**, and locate the Content Platform Engine deployment, for example `dbamc-cpe-ibm-dba-contentservices`. Click **Launch HTTP** to bring up the URL that displays the server IP and port number.

       Use these values for the subsequent status checking steps.

b) Check the Content Platform Engine Startup Context (Ping Page).

   Tip: Use a new private browser session to access the Content Platform Engine.

   Browse to `http://`*`server`*`:`*`port`*`/FileNet/Engine`, where:

   - *server* is the host name of the machine where Content Platform Engine is deployed.
   - *port* is the HTTP port where Content Platform Engine is deployed.

c) Check the automatic upgrade status for Content Platform Engine data.

   1) Enter the URL for the automatic upgrade status page into your browser.`http://`*`server_name`*`:`*`port`*`/FileNet/AutomaticUpgradeStatus`
   2) Review the status for the FileNet® P8 domain and object stores.

20. If Content Search Services was part of your container environment, start the Business Automation Configuration Container, create a new deployment, and choose only the CSS container to deploy.

    a) Add all of the same configuration details from your original configuration.

    b) Do not enable the Initialization and Verification options.

    c) Deploy the container.

    d) Check the Deployments page to see when the deployment has completed.

21. Verify your Content Search Services upgrade by using the following instructions: Verifying the IBM Content Search Services upgrade.

    Note: Text Search servers are defined in the domain **Global Configuration** > **Administrative** > **Text Search Servers** area. Confirm that the server details are accurate after the deployment. For Content Search Services, set the **Host name** and **Port values** to the IBM Cloud™ Private pod IP and the CSSssl port as seen from IBM Cloud Private. By default the CSSssl port value is 8199.

22. If Content Management Interoperability Services was part of your container environment, start the Business Automation Configuration Container, create a new deployment, and choose only the CMIS container to deploy.

    a) Add all of the same configuration details from your original configuration.

    b) Do not enable the Initialization and Verification options.

    c) Deploy the container.

    d) Check the Deployments page to see when the deployment has completed.

23. If IBM Content Navigator was part of your original container environment, start the Business Automation Configuration Container, create a new deployment, and choose only the IBM Content Navigator container to deploy.

    a) Add all of the same configuration details from your original configuration.

       Note that the IBM Content Navigator configuration container requires new values related to the Aspera configuration, as well as a new user.

    b) Do not enable the Initialization and Verification options.

    c) Deploy the container.

    d) Check the Deployments page to see when the deployment has completed.

24. Check status of IBM Content Navigator:

    Verify that you can access IBM Content Navigator at the following URL: `http://hostname:icn_port/navigator`

    Verify that the Content Platform Engine server information is correct.

    a. Log in to the Navigator Admin desktop: `http://<icp_ip_address>:<navigator_port>/navigator/?desktop=admin`
    b. Navigate to **Repositories**.
    c. Select the repository, and click **Test Connection** to verify the communication with the Content Platform Engine services are functional.

If the test fails, click **Edit** and change the **Server URL** to update the new *icp_ip_address* and *CPE_port* values:

```
icp_ip_address:CPE_port\\
```

Upgrading to V5.5.3 on certified Kubernetes

To upgrade your container deployments to V5.5.3, you must upgrade the Helm releases with the new 5.5.3 container images.

Before you begin

1. Plan your upgrade for a time when it is convenient to restrict all user access to content services.
2. If you have not already done so, make a backup of your environment folders and volumes on certified Kubernetes.
3. Download the new IBM Passport Advantage (PPA) archive and load the container images to a registry in your Kubernetes platform. After you load the images, you can view them in your Docker Registry.
4. Create a new persistent volume and persistent volume claim for the Content Search Services custom store, /css/customstore.

   Set the folder ownership to 50001:50000:

   ```
   chown -Rf 50001:50000 /css/customstore
   ```

   The following examples illustrate the YAML file contents to create a persistent volume and persistent volume claim for the Content Search Services custom store.

   Persistent volume:

   ```
   apiVersion: v1
   {
     "kind": "PersistentVolume",
     "apiVersion": "v1",
     "metadata": {
       "name": "css-icp-customstore-pv",
       "labels": {}
     },
     "spec": {
       "storageClassName": "css-icp-customstore",
       "capacity": {
         "storage": "1Gi"
       },
       "accessModes": [
         "ReadWriteMany"
       ],
       "persistentVolumeReclaimPolicy": "Retain",
       "nfs": {
         "path": "/mnt/nfsshare/css/customstore",
         "server": "<NFS_SERVER>"
       }
     }
   }
   ```

   Persistent volume claim:

   ```
   apiVersion: v1
   {
     "kind": "PersistentVolumeClaim",
     "apiVersion": "v1",
     "metadata": {
       "name": "css-icp-customstore",
       "namespace": "default"
     },
     "spec": {
       "storageClassName": "css-icp-customstore",
       "resources": {
         "requests": {
           "storage": "1Gi"
         }
       },
   ```

```
      "accessModes": [
        "ReadWriteMany"
      ]
    }
  }
```

5. (Upgrades from V5.5.2) Update the permissions on the Content Management Interoperability Services folders:

   For each of the folders, set the ownership to 50001:50000:

   ```
   chown –Rf 50001:50000 /cmiscfgstore
   ```

6. (Upgrades fromV5.5.1) If you are deploying the IBM Content Navigator container, create the persistent volume and persistent volume claim for Aspera.

   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-icp-asperastore-pv
   spec:
     accessModes:
     - ReadWriteMany
     capacity:
       storage: 1Gi
     nfs:
       path: /home/cfgstore/icn/aspera
       server: <NFS_SERVER>
     persistentVolumeReclaimPolicy: Retain
     storageClassName: icn-icp-asperastore-pv
   ```

   Persistent volume claim:

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

7. (Upgrades from V5.5.1) Update the permissions on all of the persistent volume folders in your environment:

   For each of the folders, set the ownership to 50001:50000:

   ```
   chown –Rf 50001:50000 /cpecfgstore
   ```

About this task

Each release on is created from a specific Helm chart. To upgrade a release that is created with a previous chart version, you extract the new helm charts and then upgrade the releases with the new container images.

The name of the container image to use to get the helm charts is listed here, along with the image tag.

- Configuration container job container: `ibacc-icp:3.0.0`

The names and tags of the new container images are listed here.

- Content Platform Engine container image: `cpe:ibm-dba-contentservices-3.0.0`
- IBM Content Navigator container image: `icn:ibm-dba-navigator-3.0.0`
- Content Search Services container image: `css:ibm-dba-contentsearch-3.0.0`
- IBM Content Management Interoperability Services container image: `cmis:ibm-dba-cscmis-1.7.0`

During the process of upgrading your container images, you must also perform the standard checks and post upgrade configuration tasks to ensure that your content services environment continues to function as it did before the upgrade.

Be sure to plan the timing of the upgrade to ensure minimal disruption to your users.

If you are using IBM Content Navigator with Content Platform Engine, coordinate the timing of your upgrade for that container to coincide with your content services upgrades. When you change permissions on the persistent volumes for the Content Platform Engine, the IBM Content Navigator application no longer has access to the Content Platform Engine. You can change the permissions on the persistent volumes at the same time for your IBM Content Navigator container, and then complete the upgrade steps for that container after you complete this upgrade procedure.

Instructions and sample files for deploying content services on Kubernetes are provided in the following GitHub repository https://github.com/ibm-ecm/container-samples/tree/5.5.3.

Procedure

To upgrade on certified Kubernetes:

1. Download the package and load the image archive as described in the readme for the GitHub repositoryreadme for the GitHub repository.
2. Use the upgrade commands described in the Helm readme for the GitHub repository.

(V5.5.2 and later) Upgrading external share

If you deployed external share as part of the available V5.5.1 iFix, you must upgrade the container to use external share with V5.5.2 or later.

Before you begin

Complete the upgrade steps for Content Platform Engine and IBM Content Navigator as described in Chapter 7, "(V5.5.2 or later) Upgrading container deployments," on page 129.

About this task

(V5.5.4) Because of the change to the operator deployment methodology, upgrades to V5.5.4 are not available. Instead, you migrate your environment to a new operator deployment model. For more information, see "Migrating to V5.5.4 with operators" on page 129.

The external share feature in a container environment is enabled by an additional services container. When you upgrade external share in a container environment, you deploy the updated version of this container.

This procedure assumes that you have already upgraded your content services containers to Content Platform Engine V5.5.2 or later and IBM Content Navigator V3.0.5 or later. You must have access to your previously configured volumes that contain the files that are relevant to your external share configuration.

Procedure

To upgrade your external share configuration:

1. Deploy the external share container.

   Use the information in the following topic: "Deploying the external share container" on page 148.
2. Configure the external LDAP directory:

   a) Navigate to the `/configDropins/override` directory for your Content Platform Engine container deployment.
   b) Copy the external LDAP XML file that you created, for example, `LDAPext.xml`, and the main `LDAP.xml` file.
   c) Paste both the `LDAP.xml` and the `LDAPext.xml` file to the `/configDropins/override` for the External Share container.
   d) Paste the copied external LDAP XML file, for example, `LDAPext.xml`, in the following locations on your upgraded environment:
      - The `/configDropins/override` for Content Platform Engine.
      - The `/configDropins/override` for IBM Content Navigator.
3. Configure a connection to the IBM Content Navigator database:
   a) Navigate to the `/configDropins/override` folder for your new IBM Content Navigator deployment, and copy the following files:
      - IBM Content Navigator database XML file (`ICNDS.xml`)
      - `JDBCDriver.xml`
      - JDBC database drivers
   b) Paste the files into the `/configDropins/override` folder for your new external share container deployment.
4. Navigate to the `/configDropins/override` folder for your previous External Share deployment, and copy the `CORS.xml` file to the `/configDropins/override` for your new External Share deployment.
5. Configure the Share plug-in in IBM Content Navigator.

   Use the information in the following topic: “Configuring the Share plug-in in IBM Content Navigator” on page 164.

Chapter 8. (V5.5.2 or later) Optional: Configuring external share for containers

You can configure your Content Platform Engine and IBM Content Navigator container deployments to enable the sharing of content with users that are external to your organization. Configuration for this feature includes deploying an additional container to enable external sharing.

Before you begin

Before you configure your container environment and deploy the external share container, you must perform the following steps:

- If you have not already done so, deploy and configure the Content Platform Engine V5.5.2 or later container and the IBM Content Navigator V3.0.5 or later container. Verify that these container deployments are working together as intended.
- If you want to use dynamic user provisioning, you must deploy Content Platform Engine V5.5.4 and the V5.5.4 External Share container. You must also deploy IBM Content Navigator V3.0.7, choosing the ICN-SSO container image for deployment.

  This feature requires all users to use an identity provider that supports OAuth 2.0 or OpenID Connect.

  – For Internal users, such as employees, this identity provider must contain the same set of users as the LDAP server that is used by Content Platform Engine in one of its directory configurations.

  – For External users (i.e. users outside your company), you can configure one or more identity providers. These identity providers must be different than the one used for Internal users.

  Google Sign In and IBM Id are examples of OAuth 2.0/OpenID Connect identity providers that can be used for External users.

  Regardless of the number of identity providers you configure for External Users, you only need one Managed User Directory configured in Content Platform Engine.
- If you plan to use the second LDAP directory model, prepare or designate an LDAP directory specifically for your external users. For details, see Configuring the external user LDAP realm.

About this task

Deploying the external share container and configuring your container environment for external share are part of a series of steps that make the external share capability available to users. The following roadmap provides a high-level view of these setup steps, and designates which steps are part of the container environment configuration:

1. **(Container environments)** Create volumes and folders for the external share container.
2. **(Container environments)** Prepare for and deploy your external share container.
3. **(Container environments) (V5.5.4 and later)** Configure Ingress access for your applications.
4. Choose how you want to configure external users, then configure authentication and user management:

   (Container environments) (V5.5.4 and later) Dynamic user provisioning with an Identity Provider (IDP)

   You designate an identity provider for external users, configure an Identity Provider to provide additional management of internal users, and configure additional parameters and files.

   (All environments) External LDAP user directory

   Configure or designate a customer-managed LDAP directory realm to manage your external users. Note that FileNet P8 Platform and IBM Content Navigator do not manage this LDAP realm. See Configuring the external user LDAP realm for additional information. **(Container environments)** Add the LDAP configuration details to the container deployment environment.

5. **(Container environments) (V5.5.3 and earlier)** Connect the external share container deployment to the IBM Content Navigator database.
6. **(Container environments)** Configure cross origin resource sharing (CORS) to enable the REST service for external content sharing.
7. (All environments) Configure external share settings on FileNet P8 Platform by using the Administration Console for Content Platform Engine. Settings in the administration console include configuring the additional LDAP directory realm for external users and properties that are specific to external content sharing. See Configuring Content Platform Engine for external sharing for additional information.
8. (All environments) Configure IBM Content Navigator to enable external shares. See Configuring external Share for additional information.
   - Enable external shares by enabling the P8 repository, and setting appropriate permissions.
   - Set up the external sharing capability for users by adding the share menu actions, creating a custom desktop for external users, and optionally customizing the email template for sharing.
   - You will also need to make IBM Content Navigator available to the external users, outside of the firewall. This task is typically done by a network administrator.
9. (All environments) Configuring additional Content Platform Engine settings:
   - Configuring the sweep policy for share (optional)
   - Customizing the email template (optional)

Creating volumes and folders for the external share container

The external share component containers a set of 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

Although the following 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.

The following steps illustrate example YAML file contents to create a persistent volume and persistent volume claim for the external share container deployment.

Procedure

To create volumes and folders for the external share container:

1. Create the `es-config-pv.yml` file for the configuration persistent volume:

```
{
  "kind": "PersistentVolume",
  "apiVersion": "v1",
  "metadata": {
    "name": "ecm-es-config-pv",
    "labels": {}
  },
  "spec": {
    "storageClassName": "ecm-es-config-pv",
    "capacity": {
      "storage": "1Gi"
    },
    "accessModes": [
      "ReadWriteMany"
    ],
    "persistentVolumeReclaimPolicy": "Retain",
    "nfs": {
      "path": "/mnt/nfsshare/extshare/config",
      "server": "<NFS_SERVER>"
    }
  }
}
```

2. Create the persistent volume by running the following command:

```
kubectl apply -f es-config-pv.yaml
```

3. Create the `es-config-pvc.yml` file for the configuration persistent volume claim:

```
{
  "kind": "PersistentVolumeClaim",
  "apiVersion": "v1",
  "metadata": {
    "name": "ecm-es-config-pvc",
    "namespace": "default"
  },
  "spec": {
    "storageClassName": "ecm-es-config-pv",
    "resources": {
      "requests": {
        "storage": "1Gi"
      }
    },
    "accessModes": [
      "ReadWriteMany"
    ]
  }
}
```

4. Create the persistent volume claim by running the following command:

```
kubectl apply -f es-config-pvc.yaml
```

5. Create the `es-log-pv.yml` file for the logging persistent volume:

```
{
  "kind": "PersistentVolume",
  "apiVersion": "v1",
  "metadata": {
    "name": "ecm-es-logs-pv",
    "labels": {}
  },
  "spec": {
    "storageClassName": "ecm-es-logs-pv",
    "capacity": {
      "storage": "1Gi"
    },
    "accessModes": [
      "ReadWriteMany"
    ],
    "persistentVolumeReclaimPolicy": "Retain",
    "nfs": {
      "path": "/mnt/nfsshare/extshare/logs",
      "server": "<NFS_SERVER>"
    }
  }
}
```

6. Create the persistent volume by running the following command:

```
kubectl apply -f es-log-pv.yaml
```

7. Create the `es-log-pvc.yml` file for the logging persistent volume claim:

```
{
  "kind": "PersistentVolumeClaim",
  "apiVersion": "v1",
  "metadata": {
    "name": "ecm-es-logs-pvc",
    "namespace": "default"
  },
  "spec": {
    "storageClassName": "ecm-es-logs-pv",
    "resources": {
      "requests": {
        "storage": "1Gi"
      }
    },
    "accessModes": [
      "ReadWriteMany"
    ]
  }
}
```

8. Create the persistent volume claim by running the following command:

```
kubectl apply -f es-log-pvc.yaml
```

9. Set permissions on all of the volumes by running the following command:

```
chown -R 50001:50000 /mnt/nfsshare/extshare
```

Deploying the external share container

Deploy the external share container on IBM Cloud Private or on an instance of certified Kubernetes where your Content Platform Engine and IBM Content Navigator containers are deployed.

Before you begin

Before you deploy this container release, ensure that you have deployed the following in your container environment:

For V5.5.4

- Content Platform Engine V5.5.4
- IBM Content Navigator V3.0.7

For V5.5.3

- Content Platform Engine V5.5.3
- IBM Content Navigator V3.0.6

For V5.5.2

- Content Platform Engine V5.5.2
- IBM Content Navigator V3.0.5

Make sure that your updated Content Platform Engine and IBM Content Navigator are functional before you deploy this external share container and complete the configuration of the external share feature.

This procedure assumes that you have completed the previous preparation tasks for deploying the external share container, including Creating volumes and folder.

This procedure assumes that you have already downloaded the package for the external share container from Passport Advantage.

(V5.5.4 and later) Deploying the external share container with an operator

Deploy the share container by using the operator method that you used for your Content Platform Engine and IBM Content Navigator deployments.

About this task

You can deploy External Share as part of the initial deployment that includes Content Platform Engine and IBM Content Navigator. Or you can deploy the container later, as an update to the custom YAML file for the FileNet Content Manager operator. Note that if you deploy External Share as an update, Content Platform Engine and IBM Content Navigator will undergo a rolling update as the operator reconciles the settings across the three containers.

Instructions and sample files for deploying the containers on Kubernetes are provided in GitHub. The following deployment options are available:

Table 21. Options for deployment with the operator

Platform	**Where to go**
Managed Red Hat OpenShift on IBM Cloud Public	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/roks/README.md
Red Hat OpenShift	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/ocp/README.md
Certified Kubernetes	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/k8s/README.md

Deploying the external share container on IBM Cloud Private

Deploy the share container on IBM Cloud Private where your Content Platform Engine and IBM Content Navigator containers are deployed.

Procedure

To deploy the external share container:

1. Save and expand the package in the IBM Cloud Private server boot master location.

   You see the following contents in the `extshare` subdirectory:

   - `extshare.tgz` - The container image
   - `ibm-dba-extshare-2.0.0.tgz` - A compressed file that contains the Helm Chart and YAML file
2. Change to the `./extshare` directory and load the container image by using the following command:

   ```
   docker load < extshare.tgz
   ```
3. Tag and push the image to the IBM Cloud Private cluster by using the following command:

   ```
   docker tag extshare:3.0.5 mycluster.icp:8500/namespace/share:tag
   ```
4. Log in to IBM Cloud Private:

   ```
   cloudctl login -a https://master_host:8443 --skip-ssl-validation -c id-mycluster-account
   docker push mycluster.icp:8500/<namespace>/extshare:<tag>
   ```
5. Expand the `ecm-ppa-ifix1-ibm-dba-extshare-2.0.0.tgz` file.
6. Edit the `./ibm-dba-extshare/values.yaml` file to match your environment, paying particular attention to capture the following settings and sections:
   - replicaCount
   - image
   - esProductionSettting
   - dataVolume sections
7. Save all changes to your YAML file.
8. Deploy the Helm chart by using the following command:

   ```
   helm install --name <release_name> ./ibm-dba-extshare --tls
   ```

   Where <*release_name*> is the name that you choose for the release, for example, ibacc-es.
9. Wait until the *release_name*-ibm-dba-extshare service is available and its pods are ready:

   `http://`*`hostname`*`:`*`extshare_port`*`/contentapi/rest/share/v1/info`

Deploying the external share container on certified Kubernetes

You use the Helm chart or YAML files in the GitHub repository to enter the deployment values for your external share container in Kubernetes.

Procedure

- Use the instructions and sample files for deploying the external share container on Kubernetes that are provided in the following GitHub repository

  https://github.com/ibm-ecm/container-samples/tree/5.5.3

(V5.5.4 and later) Configuring Ingress access for external share

For external share, normal operator routes can be problematic. Ingress provides access to all external share services under a single hostname.

About this task

After you deploy Content Platform Engine, IBM Content Navigator, and External Share with the Operator, you enable Ingress so that External Share can receive the appropriate cookies to authenticate between applications.

To enable Ingress, you create a YAML file that contains hostname, services, and paths.

Procedure

To enable Ingress for External Share:

1. Use the following command to retrieve the operator route for your external share deployment:

   ```
   oc get route <route-name> -o yaml
   ```

   <route-name> is the deployment name and component suffix, for example, `fncmdeploy-es-route`.
2. From the output, save the certificate key in a file called `tls.key`.
3. Also from the output, save the certificate in a file called `tls.crt`.
4. Use the files in the following command to create a Kubernetes tls secret:

   ```
   kubectl create secret tls <secret-name-tls> --key tls.key  --cert tls.crt
   ```

5. Use the following command to retrieve the infrastructure IP information:

   ```
   oc get nodes -o wide
   ```

   Save the information to use in your Ingress YAML file.
6. Create the Ingress YAML file:

   ```
   ---
   apiVersion: extensions/v1beta1
   kind: Ingress
   metadata:
    name: <ingress-name>
    annotations:
      kubernetes.io/ingress.class: "nginx"
      # The NGINX ingress annotations contains a new prefix nginx.ingress.kubernetes.io.
      nginx.ingress.kubernetes.io/affinity: "cookie"
      nginx.ingress.kubernetes.io/session-cookie-name: "route"
      nginx.ingress.kubernetes.io/secure-backends: "true"
      nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
      nginx.ingress.kubernetes.io/session-cookie-hash: "sha1"
   spec:
    rules:
    - host: <hostname.ingress>
      http:
        paths:

        - backend:
   ```

```
          serviceName: <change-me>-cpe-svc
          servicePort: http
        path: /P8CE/Health
      - backend:
          serviceName: <change-me>-cpe-svc
          servicePort: http
        path: /FileNet
      - backend:
          serviceName: <change-me>-cpe-svc
          servicePort: http
        path: /wsi
      - backend:
          serviceName: <change-me>-navigator-svc
          servicePort: http
        path: /navigator
      - backend:
          serviceName: <change-me>-es-svc
          servicePort: http
        path: /contentapi
  tls:
  - hosts:
    - <hostname.ingress>
    secretName: <secret-name-tls>
```

7. In the YAML file, update values for the following elements:

   - <secret-name-tls> The name of the secret that you created in step 4.
   - <hostname.ingress> The hostname you want to use for Ingress. Choose a valid top-level domain name, like `hostname.ingress.ibm.com`.
   - <change-me> The prefix for your services that the operator created.
   - <ingress-name> The name of the Ingress deployment.

   For deployments using two identity providers (instead of 2 LDAP directories), additional configuration is required in the `ingress.yml` file. You must add `-backend` rules for each identity provider that you use. Use the unique ID for your Open ID Client that you configured in the `zoidc.xml` file, for example, ExShareGIDCPE for Content Platform Engine, ExShareGIDES for External Share, and ExShareGIDICN for IBM Content Navigator.

   Add these additional paths to the backend rules. The following example shows Google Sign In settings:

```
     - backend:
          serviceName: <change-me>-cpe-svc
          servicePort: http
       path: /oidcclient/redirect/ExShareGIDCPE
     - backend:
          serviceName: <change-me>-navigator-svc
          servicePort: http
       path: /oidcclient/redirect/ExShareGIDICN
     - backend:
          serviceName: <change-me>-es-svc
          servicePort: http
       path: /oidcclient/redirect/ExShareGIDES
```

   For UMS, the additional Ingress elements might look like the following example:

```
     - backend:
          serviceName: <change-me>-cpe-svc
          servicePort: http
       path: /oidcclient/redirect/cpe
     - backend:
          serviceName: <change-me>-navigator-svc
          servicePort: http
           path: /oidcclient/redirect/navigator
     - backend:
          servicePort: http
           path: /oidcclient/redirect/extshare
```

8. Use the following command to create a new .p12 file:

```
openssl pkcs12 -export -out file_to_generate.p12 -inkey tls.key -in tls.crt
```

9. Put a copy of the .p12 file that you generated in the `confiDropins/overrides` directory for Content Platform Engine, IBM Content Navigator, and External Share.
10. For the Content Platform Engine deployment, create a copy of the `ibm_custom-ssl.xml` from the container, name it `zibm_custom-ssl.xml` and move it to the `configDropins/overrides` directory:

    ```
    oc exec -it <pod_name> cp /opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
    ibm_custom-ssl.xml /opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
    zibm_custom-ssl.xml
    ```

    For example:

    ```
    <?xml version="1.0" encoding="UTF-8"?>
    <server>
        <sslDefault sslRef="cssSSLSettings"/>
        <!-- SSL keystore and truststore configuration for FNCM -->
        <ssl id="cssSSLSettings"
            keyStoreRef="customKeyStore"
            trustStoreRef="customTrustStoreNEW"
            clientAuthenticationSupported="false"
            sslProtocol="TLSv1.2"
            securityLevel="CUSTOM"
            enabledCiphers="TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
    TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
    TLS_DHE_RSA_WITH_AES_128_GCM_SHA256"
            />
        <keyStore id="customKeyStore"
            location="ibm_customFNCMKeyStore.p12"
            type="PKCS12" password="xxxxxxxx" />
        <keyStore id="customTrustStore"
            location="ibm_customFNCMTrustStore.p12"
            type="PKCS12" password="xxxxxxxx" />

        <keyStore id="customTrustStoreNEW"
            location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
    new_custom_tls.p12"
            type="PKCS12" password="xxxxxx=" />
    </server>
    ```

11. Repeat the previous step for IBM Content Navigator and for External Share.
12. Confirm that your setting for `customTrustStoreNEW` points to the location of the XML override file that you just created:

    ```
    <keyStore id="customTrustStoreNEW"
            location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
    new_custom_tls.p12"
            type="PKCS12" password="xxxxxxxxx" />
    ```

13. On the platform, edit the deployments for IBM Content Navigator and External share to include the hostname.
    a) Run the following command with the value for your namespace to get the deployment names:

    ```
    oc get deployments -n <namespace>
    ```

    b) Open the IBM Content Navigator deployment for editing:

    ```
    oc edit deployment -n <namespace> <deployment-name-navigator>
    ```

    c) Under `spec.template.spec`, make the following update:

    ```
            hostAliases:
            - ip: "INFRASTRUCTURE_IP"
              hostnames:
              - "<hostname.ingress>"
    ```

    d) Repeat the steps to open and edit the External Share deployment with the same update.

    After you save and close the deployment, the pods are recreated with the new changes in place.
14. Apply the Ingress YAML to your namespace:

```
oc apply -f ingress.yaml
```

15. Update the DNS records to include the Ingress hostname, for example, `hostname.ingress.ibm.com`.

    As a result, access for users is granted to the applications with URLs like the following:

    `https://<operator_route>/acce`

    `https://hostname.ingress.ibm.com/navigator`

    A URL with this Ingress hostname is sent to external users for access to shared content.

Configuring external share users

The steps you take to configure your external users vary depending on the method you want to use. You can manage users with a second LDAP directory for external users, or you can use an Identity Provider (IDP).

About this task

External share offers two ways to manage external access to content:

- Using a separate LDAP realm for external users
- (V5.5.4 and later)Dynamic user provisioning with an Identity Provider (IDP), where internal and external users are configured in separate identity providers(IDP) that support both OAuth 2.0 and/or OpenID connect.

These approaches provide different benefits and require different steps to configure and manage:

(V5.5.4) Dynamic user provisioning

With dynamic external user provisioning, you can grant access to external user that have accounts with external identity provides (IDPs) like IBM ID or Google ID. When an internal user wants to grant access to the external user, the email is entered into a managed user directory on the Content Platform Engine on a provisional basis. When the user responds to the share invitation, the managed user record is confirmed.

This method for managing access and users requires additional configuration and setup:

- Designated internal user IDP, such as UMS. Where the same set of users resides in an LDAP used by the Content Platform Engine which is accessed directly.
- Designated external user IDP. External users are identified by unique email addresses that must not also exist in the LDAP used for internal users.
- Setup Managed User Directory configuration for external IDP users.
- Additional configuration files and parameters for the external share environment
- The ICN-SSO container image for your IBM Content Navigator deployment

External user LDAP

Managing external users with a separate external LDAP directory requires that you configure a second LDAP directory to manage the external users who can share content. You can use any of the supported LDAP directory service providers to configure the external user directory. Note that because of the need for two separate and distinct directory configurations, IBM Virtual Member Manager is not supported.

You must designate and add these external users to the LDAP directory before internal users can issue share invitations to the external users.

(V5.5.4 and later) Configuring users with an Identity Provider (IDP)

Dynamic user provisioning requires you to configure at least two Identity Provider instances, one for the internal users and one or more for the external users. The Internal users are configured with an IDP that contains the same set of users as the LDAP server used by the Content Platform Engine in one of its directory configurations.

Before you begin

Set up the Managed User Directory configuration using the Administration Console for Content Platform Engine. Configure only one Managed User directory for the external users, even if more than one IDP is configured for the external users. The external users must be uniquely identified by the email address across all realms specified in the P8 domain, other than the Managed User Directory realm. To configure the Managed User Directory configuration see Configuring users with an identity provider.

All Identity Providers (IdPs) that support OAuth 2.0 or OpenID Connect authentication have a registration mechanism to identify the client application to the Identity Provider. At a minimum, a client ID, client secret, and redirect URLs to the client application are required by the OAuth 2.0 and OpenID Connect specifications.

For an example, you can review the Google Identity Platform documentation that describes how to obtain OAuth 2.0 credentials for applications: OpenID Connect.

When registering an application with an Identity Provider, you can use the same clientId registration for each of the IBM Content Navigator, External Share, and Content Platform Engine instances in your environment. You must provide a redirect URL for each of these instances using the following pattern:

https://<*ingress_host*>/oidcclient/redirect/<*ExShareId*>

Where:

<*ingress_host*> is the host name specified in the Ingress YAML file for configuring your IBM Content Navigator, External Share, and Content Platform Engine instance. For more information, see "(V5.5.4 and later) Configuring Ingress access for external share" on page 150.

<*ExShareId*> is the ID that is used in the openIdConnectClient configuration for your IBM Content Navigator, External Share, and Content Platform Engine instance in the configuration file, for example, `zoidc.xml`. This ID is used in the redirection URI provided to the Identity Provider to configure the client ID redirection. For example, openidConnectClient id configured as ExShareGIDCPE for Content Platform Engine, ExShareGIDES for External Share, and ExShareGIDICN for IBM Content Navigator.

Each deployment requires a unique URI for redirection back from the Identity Provider to distinguish itself for the appropriate service. The URI are entered at the Identity providers registration page.

As an example, for Google Sign In Identity provider, the OAuth 2.0 client ID for ExShareGID would have three Authorized redirect URIs entered by the user, one for each deployment:

https://hostname.ingress.ibm.com/oidcclient/redirect/ExShareGIDCPE

https://hostname.ingress.ibm.com/oidcclient/redirect/ExShareGIDES

https://hostname.ingress.ibm.com/oidcclient/redirect/ExShareGIDICN

About this task

To prepare for managing users with an external and internal identity provider, you must configure your environment to communicate securely with the identity providers. Note that most configuration steps are repeated for each component container, IBM Content Navigator, External Share, and Content Platform Engine.

For an example, the following steps assume that you are using UMS for the internal identity provider, and Google Sign In for your external user identity provider.

Note: For V5.5.4, name any files in the override directory with a z prefix. This ensures a file order that means that your version of the file, for example, `zoidc.xml` will override any other version.

Procedure

To configure user management with an Identity Provider (IDP):

1. Import the Identity Provider SSL certificate into the application's (for example, IBM Content Navigator's) trust store:
   a) Obtain the identity provider's SSL certificate.

      The following example shows a command that you might use with a UMS or other WebSphere Liberty based identity provider:

      ```
      keytool -exportcert -keystore key.p12 -storetype pkcs12 -alias default -rfc -file
      umscert.pem
      ```

      Google uses GlobalSign as the certificate authority for Google Sign In. You can obtain the certificate from your browser at https://accounts.google.com or download it from a site such as https://www.tbs-certificates.co.uk/FAQ/en/GlobalSign_Root_CA_R2_root.html
   b) Import this certificate in the IBM Content Navigator's trust store, giving it a unique alias name.

      Use the trust store that you created when you configured Ingress, for example, that you defined in `zibm_custom-ssl.xml`.

      The following examples show commands for importing your identity provider certificates into your custom trust store.

      For UMS:

      ```
      keytool -importcert -trustcacerts -keystore new_custom_tls.p12 -storetype pkcs12 -
      storepass PASSWORD -alias ums -file umscert.pem
      ```

      For Google:

      ```
      keytool -import -noprompt -keystore new_custom_tls.p12 -file globalsignrootca.crt -
      storepass changeit -v -alias gmailglobalsignrootca -storetype pkcs12
      ```
   c) Repeat these steps for the External Share and Content Platform Engine configurations.
2. Configure JVM options for the IBM Content Navigator.
   a) Create the file `${server.config.dir}/configDropins/overrides/jvm.options` with the following content:

      ```
      -Dcom.filenet.authentication.providers=ExShareUmsInternal,ExShareGID
      -Dcom.filenet.authentication.ExShareUmsInternal.displayname=Employee Login
      -Dcom.filenet.authentication.ExShareGID.displayname=Google Sign-In
      -Dcom.filenet.authentication.ExShareGID.AuthTokenOrder=oidc,oauth,ltpa
      ```

      Notes:

      - `com.filenet.authentication.providers`

        This is a comma separated list of identity provider identifiers. There should be at least 2 identity providers: one for internal users (e.g. your employees) and one for external users.
      - `com.filenet.authentication.<providerid>.displayname`

        Using the <providerid> listed under `com.filenet.authentication.providers`, specify the text of the login button associated with that identity provider.
      - `com.filenet.authentication.<providerid>.AuthTokenOrder=oidc,oauth,ltpa`

        This option allows a different order to be applied for a specific identity provider when determining the authentication token type to be propagated to the Content Platform Engine server. For example, Google authentication requires that the OpenID Connect id_token is propagated to the Content Platform Engine server for SSO. Hence the `oidc` option is listed first, followed by `oauth`, then `ltpa` in the example above. These are the 3 authentication token types supported by the Content Platform Engine server.
3. Configure Open Id Connect on the IBM Content Navigator:
   a) Create a file, for example `zoidc.xml`, in the `configDropins/overrides` directory that you created for the IBM Content Navigator deployment.

You can start with the sample `zoidc.xml` file that is provided on the git hub location. Your file must include the OpenID Client Connect information for both the internal and external identity providers, including the feature for Open Id Client Connect 1.0, an `authFilter` parameter for each identity provider, and the Open ID Client Connect values for each identity provider:

```
<?xml version='1.0' encoding='UTF-8'?>
<server>
    <featureManager>
        <feature>openidConnectClient-1.0</feature>
    </featureManager>

    <webAppSecurity ssoCookieName="FileNetLtpaToken" overrideHttpAuthMethod="CLIENT_CERT"
allowAuthenticationFailOverToAuthMethod="FORM" loginFormURL="/navigator/idplogin.jsp"
loginErrorURL="/navigator/loginError.jsp" />

    <logging
traceSpecification="com.ibm.ws.webcontainer.*=all:com.ibm.webcontainer.security.*=all:com.
ibm.ws.security.*=all:com.ibm.websphere.security.*=all"
        consoleLogLevel="INFO"
        traceFileName="trace.log"
        maxFileSize="200"
        maxFiles="10"
        traceFormat="BASIC"/>

    <authFilter id="ExShareUmsInternalFilter">
        <cookie id="ExShareUmsInternalCookie" name="ExShareUmsInternal"
matchType="contains"/>
        <cookie id="ltpatokencookie" name="FileNetLtpaToken" matchType="notContain"/>
    </authFilter>

    <openidConnectClient id="navigator"
        authFilterRef="ExShareUmsInternalFilter"
        clientId="navigator"
        clientSecret="xxxxxxxx"
        audiences="cpe,navigator,extshare"
        realmName="ExShareUmsInternal"
        userIdentifier="sub"
        uniqueUserIdentifier="uniqueSecurityName"
        userIdentityToCreateSubject="uniqueSecurityName"
        discoveryEndpointUrl="https://server_name:port/oidc/endpoint/ums/.well-known/
openid-configuration"
        mapIdentityToRegistryUser="false"
        scope="openid email profile"
        responseType="id_token token"
        httpsRequired="true"
        tokenReuse="true"
    inboundPropagation="supported"
    signatureAlgorithm="RS256"
        authnSessionDisabled="false"
        disableLtpaCookie="false"
        validationMethod="introspect">
    </openidConnectClient>

    <authFilter id="ExShareGIDFilter">
        <cookie id="ExShareGIDCookie" name="ExShareGID" matchType="contains"/>
        <cookie id="ltpatokencookie" name="FileNetLtpaToken" matchType="notContain"/>
    </authFilter>

    <openidConnectClient id="ExShareGIDICN"
        authFilterRef="ExShareGIDFilter"
        clientId="xxxxxxxxx-xxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com"
        clientSecret="xxxxxxxxxxxxxxxxxxxx"
        audiences="xxxxxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com"
        realmName="ExShareGIDICN"
        userIdentifier="email"
        uniqueUserIdentifier="email"
        userIdentityToCreateSubject="email"
        tokenReuse="true"
        discoveryEndpointUrl="https://accounts.google.com/.well-known/openid-
configuration"
        mapIdentityToRegistryUser="false"
        scope="openid email profile"
        responseType="code"
        inboundPropagation="supported"
        signatureAlgorithm="RS256"
        authnSessionDisabled="false"
        disableLtpaCookie="true"
        validationMethod="introspect"
        issuerIdentifier="https://accounts.google.com"
        jwkClientId="xxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com"
        jwkClientSecret="xxxxxxxxxxxxxx"
        jwkEndpointUrl="https://www.googleapis.com/oauth2/v3/certs"
        discoveryEndpointUrl=https://accounts.google.com/.well-known/openid-configuration
        userIdentifier="email"
        uniqueUserIdentifier="email"
        userIdentityToCreateSubject="email”>
    </openidConnectClient>

</server>
```

b) Confirm or set your values for the internal and external `authFilter` parameters.

The `authFilter` for IBM Content Navigator uses a cookie. This cookie value is set when the user clicks the button on the login dialog that corresponds to identity provider. In the `ExShareUmsInternalFilter` `authFilter`, the name given for cookie `ExShareUmsInternalCookie` must match the corresponding `providerid` specified in the `jvm.options` (i.e. `ExShareUmsInternal`).

Likewise, for the `ExShareGIDFilter` `authFilter`, the name given for cookie ExShareGIDCookie must match the corresponding `providerid` specified in the `jvm.options` (i.e. `ExShareGID`).

c) Confirm or set your values for the `openidConnectClient` parameters.

The value for id, for example, "navigator", must match the target of a redirect_uri in the registration with the identity provider. For example,

https://icn-host:9443/oidcclient/redirect/ExShareUms

The value for `realmName` must match the corresponding `providerid` specified in `jvm.options`. For example in the `navigator` configuration, the `realmName` property is set to `ExShareUmsInternal`.

The value for audiences must include your `clientId` plus the `clientId` of other applications that you interact with. For example in the `navigator` configuration, the audiences property is set to `cpe,navigator,extshare`. Depending on your identity provider, additional configuration might be required to support this scenario. For more details, refer to Configuring an OpenID Connect Client to use the RSA-SHA256 algorithm for signing ID tokens.

d) Update the values for the `clientId` and `clientSecret` for the external Identity Provider.

The following example shows a sample format for values for a Google ID configuration:

```
clientId="xxxxxxxxxxxxxxxx.apps.googleusercontent.com"
clientSecret="xxxxxxxxxxxxxxxxxxxxxxxx"
audiences="xxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com”
```

e) Update other values that are specific to your external Identity Provider.

f) Add values to override the web.xml:

```

    <webAppSecurity ssoCookieName="FileNetLtpaToken" overrideHttpAuthMethod="CLIENT_CERT"
allowAuthenticationFailOverToAuthMethod="FORM" loginFormURL="/navigator/idplogin.jsp"
loginErrorURL="/navigator/loginError.jsp" />
```

g) Save all changes to the `zoidc.xml` file.

4. Configure JVM options for External Share:

a) Create the file `${server.config.dir}/configDropins/overrides/jvm.options` with the following content:

```

-Dcom.filenet.authentication.ExShareGID.AuthTokenOrder=oidc,oauth,ltpa
```

Notes:

`com.filenet.authentication.<providerid>.AuthTokenOrder=oidc,oauth,ltpa`

This option allows a different order to be applied for a specific identity provider when determining the authentication token type to be propagated to the Content Platform Engine server. For example, Google authentication requires that the OpenID Connect `id_token` is propagated to the

Content Platform Engine server for SSO. Hence the `oidc` option is listed first, followed by `oauth`, then `ltpa` in the example above. These are the 3 authentication token types supported by the Content Platform Engine server. The <providerid> comes from the realm associated to the inbound SSO token. This realm is set by the sender of the SSO token, for example, IBM Content Navigator, via the `realmName` attribute of the sender's `openidConnectClient` configuration.

5. Configure Open Id Connect for External Share:

   a) Create a file called `zoidc.xml` in the `configDropins/overrides` directory that you created for the External Share deployment.

   You can start with the sample `zoidc.xml` file that is provided on the git hub location:

```
<?xml version='1.0' encoding='UTF-8'?>
<server>
    <featureManager>
        <feature>openidConnectClient-1.0</feature>
    </featureManager>

    <webAppSecurity ssoCookieName="FileNetLtpaToken" overrideHttpAuthMethod="CLIENT_CERT"
allowAuthenticationFailOverToAuthMethod="BASIC"/>

    <authFilter id="ExShareUmsInternalFilter">
        <requestHeader id="ExShareUmsInternalHeader" name="auth-token-realm"
value="ExShareUmsInternal" matchType="contains"/>
        <cookie id="ltpatokencookie" name="FileNetLtpaToken" matchType="notContain"/>
    </authFilter>

    <openidConnectClient id="extshare"
        authFilterRef="ExShareUmsInternalFilter"
        clientId="extshare"
        clientSecret="xxxxxxxxx"
        audiences="cpe,navigator,extshare"
        realmName="ExShareUmsInternal"
        userIdentifier="sub"
        uniqueUserIdentifier="uniqueSecurityName"
        userIdentityToCreateSubject="uniqueSecurityName"
        discoveryEndpointUrl="https://cpe-ums1.fyre.ibm.com:9443/oidc/endpoint/ums/.well-
known/openid-configuration"
        mapIdentityToRegistryUser="false"
        scope="openid email profile"
        responseType="code"
        inboundPropagation="supported"
        signatureAlgorithm="RS256"
        authnSessionDisabled="false"
        disableLtpaCookie="true"
        validationMethod="introspect">
    </openidConnectClient>

    <authFilter id="ExShareGIDFilter">
        <requestHeader id="ExShareGIDHeader" name="auth-token-realm" value="ExShareGID"
matchType="contains"/>
        <cookie id="ltpatokencookie" name="FileNetLtpaToken" matchType="notContain"/>
    </authFilter>

    <openidConnectClient id="ExShareGIDES"
        authFilterRef="ExShareGIDFilter"
        clientId="xxxx.apps.googleusercontent.com"
        clientSecret="xxxxxx"
        audiences="xxxx.apps.googleusercontent.com"
        realmName="ExShareGID"
        userIdentifier="email"
        uniqueUserIdentifier="email"
        userIdentityToCreateSubject="email"
        tokenReuse="true"
        discoveryEndpointUrl="https://accounts.google.com/.well-known/openid-
configuration"
        mapIdentityToRegistryUser="false"
        scope="openid email profile"
        responseType="code"
        inboundPropagation="supported"
        signatureAlgorithm="RS256"
        authnSessionDisabled="false"
        disableLtpaCookie="true"
        validationMethod="introspect"
        issuerIdentifier="https://accounts.google.com"
        jwkClientId="xxxxxxxxxx-xxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com"
        jwkClientSecret="xxxxxxxxxxxxxxxxxxx"
        jwkEndpointUrl="https://www.googleapis.com/oauth2/v3/certs"
        discoveryEndpointUrl=https://accounts.google.com/.well-known/openid-configuration
        userIdentifier="email"
        uniqueUserIdentifier="email"
        userIdentityToCreateSubject="email”>
    </openidConnectClient>

</server>
```

b) Confirm or set your values for the internal and external `authFilter` parameters.

The `authFilter` for External Share uses a HTTP request header with name of "auth-token-realm" and value corresponding to the identity provider. In the example, the value is `ExShareUmsInternal` for the identity provider used for internal users and is `ExShareGID` for the Google Sign In identity provider used for external users.

c) Confirm or set your values for the `openidConnectClient` parameters.

- The value for id, for example, "extshare", must match the target of a redirect_uri in the registration with the identity provider. For example,

  https://es-host:9443/oidcclient/redirect/ExShareUms
- The value for `realmName` must match the corresponding `providerid` specified in `jvm.options`. For example in the `ExShareUms` configuration, the `realmName` property is set to `ExShareUmsInternal`.
- The value for audiences must include your `clientId` plus the `clientId` of other applications that might invoke this application with an OpenID Connect token. For example in the `extshare` configuration, the audiences property is set to `cpe,navigator,extshare`. Depending on your identity provider, additional configuration might be required to support this scenario. For more details, refer to Configuring an OpenID Connect Client to use the RSA-SHA256 algorithm for signing ID tokens.

d) Update the values for the `clientId` and `clientSecret` for the external Identity Provider.

The following example shows a sample format for values for a Google ID configuration:

```
clientId="xxxxxxxxxxxxxxxx.apps.googleusercontent.com"
clientSecret="xxxxxxxxxxxxxxxxxxxxxx"
audiences="xxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com”
```

e) Update other values that are specific to your external Identity Provider.

f) Add values to override the web.xml:

```
 <webAppSecurity ssoCookieName="FileNetLtpaToken" overrideHttpAuthMethod="CLIENT_CERT"
allowAuthenticationFailOverToAuthMethod="BASIC"/>
```

g) Save all changes to the `zoidc.xml` file.

6. Configure Open Id Connect on the Content Platform Engine:

a) Create a file called `zoidc.xml` in the `overrides` directory that you created for the Content Platform Engine deployment.

You can start with the sample `zoidc.xml` file that is provided on the git hub location:

```
<?xml version='1.0' encoding='UTF-8'?>
<server>
    <featureManager>
        <feature>openidConnectClient-1.0</feature>
    </featureManager>

    <authFilter id="ExShareUmsInternalFilter">
        <requestHeader id="ExShareUmsInternalHeader" name="auth-token-realm"
value="ExShareUmsInternal" matchType="contains"/>
    </authFilter>

    <openidConnectClient id="cpe"
        authFilterRef="ExShareUmsInternalFilter"
        clientId="cpe"
        clientSecret="xxxxxxx"
        audiences="cpe,navigator,extshare"
        realmName="ExShareUmsInternal"
        userIdentifier="sub"
        uniqueUserIdentifier="uniqueSecurityName"
        userIdentityToCreateSubject="uniqueSecurityName"
        discoveryEndpointUrl="https://cpe-ums1.fyre.ibm.com:9443/oidc/endpoint/ums/.well-
known/openid-configuration"
        mapIdentityToRegistryUser="false"
        scope="openid email profile"
        responseType="code"
        inboundPropagation="supported"
        signatureAlgorithm="RS256"
        authnSessionDisabled="false"
        validationMethod="introspect">
    </openidConnectClient>

    <authFilter id="ExShareGIDFilter">
        <requestHeader id="ExShareGIDHeader" name="auth-token-realm" value="ExShareGID"
matchType="contains"/>
    </authFilter>

    <openidConnectClient id="ExShareGIDCPE"
        authFilterRef="ExShareGIDFilter"
        clientId="530122881973-
fuotgltih4t5e3335im9aeca2uql7q52.apps.googleusercontent.com"
        clientSecret="YPcdr1FifclLuF2Dyu164WWD"
        audiences="530122881973-
fuotgltih4t5e3335im9aeca2uql7q52.apps.googleusercontent.com"
        realmName="ExShareGID"
        userIdentifier="email"
        uniqueUserIdentifier="email"
        userIdentityToCreateSubject="email"
        tokenReuse="true"
        discoveryEndpointUrl="https://accounts.google.com/.well-known/openid-
configuration"
        mapIdentityToRegistryUser="false"
        scope="openid email profile"
        responseType="code"
        inboundPropagation="supported"
        signatureAlgorithm="RS256"
        authnSessionDisabled="false"
        validationMethod="introspect"
        issuerIdentifier="https://accounts.google.com"
        jwkClientId="530122881973-
fuotgltih4t5e3335im9aeca2uql7q52.apps.googleusercontent.com"
        jwkClientSecret="YPcdr1FifclLuF2Dyu164WWD"
        jwkEndpointUrl="https://www.googleapis.com/oauth2/v3/certs"
        discoveryEndpointUrl=https://accounts.google.com/.well-known/openid-configuration
        userIdentifier="email"
        uniqueUserIdentifier="email"
        userIdentityToCreateSubject="email”>
    </openidConnectClient>

</server>
```

b) Confirm or set your values for the internal and external `authFilter` parameters.

The `authFilter` for Content Platform Engine uses an HTTP request header with name of "auth-token-realm" and value corresponding to the identity provider. In the example, the value is `ExShareUmsInternalfor` the identity provider used for internal users and is `ExShareGID` for the Google Sign In identity provider used for external users.

c) Confirm or set your values for the `openidConnectClient` parameters.

- The value for id, for example, "cpe", must match the target of a redirect_uri in the registration with the identity provider. For example,

  `https://cpe-host:9443/oidcclient/redirect/cpe`

- The value for `realmName` must match the corresponding `providerid` associated with the inbound SSO token. For example in the `cpe` configuration, the `realmName` property is set to `ExShareUmsInternal`.
- The value for audiences must include your `clientId` plus the `clientId` of other applications that you interact with. For example in the `cpe` configuration, the audiences property is set to `cpe,navigator,extshare`. Depending on your identity provider, additional configuration might be required to support this scenario. For more details, refer to Configuring an OpenID Connect Client to use the RSA-SHA256 algorithm for signing ID tokens

d) Update the values for the `clientId` and `clientSecret` for the external Identity Provider.

The following example shows a sample format for values for a Google ID configuration:

```
clientId="xxxxxxxxxxxxxxxx.apps.googleusercontent.com"
clientSecret="xxxxxxxxxxxxxxxxxxxxxxx"
audiences="xxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxx.apps.googleusercontent.com”
```

e) Update other values that are specific to your external Identity Provider.

f) Save all changes to the `zoidc.xml` file.

Configuring the external user LDAP realm

In the container environment, before you can you add the external user LDAP server in the Administration Console for Content Platform Engine and in IBM Content Navigator, you add the external server as an additional `LDAP.xml` file in the configuration overrides directory for each component.

Before you begin

The steps and requirements for setting up the external-user LDAP directory server are the same for on-premises and container environments. Set up your external LDAP server per the instructions in Configuring the external user LDAP realm, then return to complete the container-specific piece of the configuration.

About this task

The method for configuring the second LDAP directory that contains external users is different depending on which version you are deploying:

V5.5.4 and later

With the operator deployment, you provide details about the external LDAP directory in the `ldap_configuration` section of the CR YAML file for the operator deployment:

```
ext_ldap_configuration:
  #   # the candidate value is "IBM Security Directory Server" or "Microsoft Active
Directory"
  #   lc_selected_ldap_type: "IBM Security Directory Server"
  #   lc_ldap_server: "<hostname>"
  #   lc_ldap_port: "389"
  #   lc_bind_secret: ldap-bind-secret # secret is expected to have ldapUsername and
ldapPassword keys
  #   lc_ldap_base_dn: "O=LOCAL"
  #   lc_ldap_ssl_enabled: false
  #   lc_ldap_ssl_secret_name: ""
  #   lc_ldap_user_name_attribute: "*:cn"
  #   lc_ldap_user_display_name_attr: "cn"
  #   lc_ldap_group_base_dn: "O=LOCAL"
  #   lc_ldap_group_name_attribute: "*:cn"
  #   lc_ldap_group_display_name_attr: "cn"
  #   lc_ldap_group_membership_search_filter: "(|(&(objectclass=groupofnames)(member={0}))
(&(objectclass=groupofuniquenames)(uniquemember={0})))"
  #   lc_ldap_group_member_id_map: "groupofnames:member"
  #   ad:
  #     lc_ad_gc_host: ""
  #     lc_ad_gc_port: ""
  #     lc_user_filter: "(&(cn=%v)(objectclass=person))"
  #     lc_group_filter: "(&(cn=%v)(|(objectclass=groupofnames)
(objectclass=groupofuniquenames)(objectclass=groupofurls)))"
  #   tds:
  #     lc_user_filter: "(&(cn=%v)(objectclass=person))"
  #     lc_group_filter: "(&(cn=%v)(|(objectclass=groupofnames)
(objectclass=groupofuniquenames)(objectclass=groupofurls)))"
```

Supply values for these parameters, along with other relevant values for your External Share deployment, before you run the deployment. Because these settings are common across components, they apply for Content Platform Engine, IBM Content Navigator, and External Share. No `LDAPext.xml` file is needed.

V5.5.3 and earlier

You set up the external LDAP configuration in the `LDAPext.xml` file. Use the steps in the following procedure.

Procedure

(V5.5.3 and earlier) To configure the external LDAP directory:

1. Navigate to the `/configDropins/override` directory for your Content Platform Engine container deployment.
2. Create a copy of the `LDAP.xml` file, for example, `LDAPext.xml`.
3. Modify the new copy of the XML file by adding the details for the external LDAP directory server that you configured specifically for external share.

   The realm name in the new file must be different from the realm name in the original file.
4. Save the updated copy of `LDAPext.xml` in the overrides directory.
5. Copy the updated `LDAPext.xml` file into the `/configDropins/override` directory for the IBM Content Navigator container deployment.
6. Copy both the original `LDAP.xml` and the new `LDAPext.xml` file into the `/configDropins/override` directory for the external share container deployment.

Results

The two LDAP XML files must be present in the `/configDropins/override` directory for Content Platform Engine, IBM Content Navigator, and for the external share container.

Important: The realm name in the `LDAPext.xml` file must be the same in each copy, and must be different from the realm name in the original `LDAP.xml` file.

(V5.5.3 and earlier) Connecting to the IBM Content Navigator database

The external share container requires access to the IBM Content Navigator database.

About this task

Use the following steps to enable the Share container to access the IBM Content Navigator database.

Procedure

To connect to the IBM Content Navigator database:

1. Copy the following items from the IBM Content Navigator <*pvc-icn-dir*>`/configDropins/override` directory:
   - IBM Content Navigator database XML file (`ICNDS.xml`)
   - `JDBCDriver.xml`
   - JDBC database drivers
2. Paste the items into the `/configDropins/override` directory for the external share container.

Configuring cross origin resource sharing (CORS)

Cross origin resource sharing (CORS) allows requests from a different URL to be processed by the REST service that is running in the share container, and allows responses to be returned.

About this task

You enable the resource sharing by updating and adding the CORS.xml file to the `/configDropins/override` directory for the external share container.

Procedure

To configure the `CORS.xml` file:

1. Copy the `CORS.xml` sample file from the following Github location:
   a. For V5.5.4: https://github.com/ibm-ecm/container-samples/tree/5.5.4/extShare
   b. For V5.5.3: https://github.com/ibm-ecm/container-samples/tree/5.5.3/extShare

2. Save the copy to the `/configDropins/override` directory for your external share container.

   Note: For V5.5.4, rename the CORS.xml in the override directory to `zCORS.xml`. This ensures a file order that means this version of the cross origin resource sharing will override any other version.

3. Edit the **`allowedOrigins`** parameter and enter the IBM Content Navigator domain name.

   The domain name consists of HTTP (non-SSL) or HTTPS (SSL), and the IBM Content Navigator host name and port that matches the requests coming from the client.

   For example (V5.5.4)

```
<server>
  <cors domain="/contentapi/rest/share/v1"
        allowedOrigins="https://hostname.ingress.ibm.com"
        allowedMethods="GET, DELETE, PUT, POST, OPTIONS"
        allowedHeaders="auth-token-realm,Host,Authorization,Sec-Fetch-Site,
        Sec-Fetch-Mode,Connection,Pragma,Cache-Control,Navigator-Client-Build,
        ECM-XSRF-Token,Origin,User-Agent,Content-Type,Content-Length,
        Navigator-Client-Identity,Accept-Control-Request-Method,Accept-Control-Request-
Headers,
        Accept,Referer,Accept-Encoding,Accept-Language,DNT,Host,Content-Length,Cache-
control,Cookie"
        exposeHeaders="Expires,Pragma,Set-Cookie,Content-Length,Content_Type,Content-
Language,
        X-Powered-By,Date,Allow,Transfer-Encoding,$WSEP,DNT,Access-Control-Allow-Credentials,
        Access-Control-Allow-Headers,Access-Control-Allow-Max-Age,Access-Control-Allow-
Methods,
        Access-Control-Allow-Origin,Access-Control-Expose-Headers,Connection,Cache-
control,Cookie"
        allowCredentials="true"
        maxAge="3600" />
</server
```

   Make sure that the allowedOrigins entry matches your Ingress configuration hostname. See "(V5.5.4 and later) Configuring Ingress access for external share" on page 150.

   For example (V5.5.3 and earlier):

```
        <cors domain="/contentapi/rest/
share/v1"

     allowedOrigins="http://
myicn:9080“

     allowedMethods="GET, DELETE, PUT, POST,
OPTIONS"

     allowedHeaders="Connection,Pragma,Cache-Control,Navigator-Client-Build,ECM-XSRF-Token,
     Origin,User-Agent,Content-Type,Content-Length,Navigator-Client-Identity,
     Accept-Control-Request-Method,Accept-Control-Request-Headers,Accept,Referer,Accept-
Encoding,Accept-Language,DNT,Host,Content-Length,Cache-control,Cookie"
     exposeHeaders="Content-Length,Content_Type,Content-Language,X-Powered-By,
     Date,Allow,Transfer-Encoding,$WSEP,DNT,Access-Control-Allow-Credentials,
     Access-Control-Allow-Headers,Access-Control-Allow-Max-Age,Access-Control-Allow-Methods,
     Access-Control-Allow-Origin,Access-Control-Expose-Headers,Connection,Cache-
control,Cookie"
     allowCredentials="true"
     maxAge="3600" />
```

   You can add both HTTP and HTTPS values for the **`allowedOrigins`** value. Use a comma to separate multiple values for the entry.

   Trailing slashes (/) are not allowed.

   Important: Do not modify any parameter values other than the **`allowedOrigins`** value.

Configuring the Share plug-in in IBM Content Navigator

In a container environment, you configure the Share plug-in JAR setting in IBM Content Navigator to point to a URL.

About this task

A setting in IBM Content Navigator directs the application to the Share plug-in. This setting is different when external share is configured in a container environment. You set this value as part of the configuration steps for external share on IBM Content Navigator. For context, see the following IBM Content Navigator topic: Configuring external share

Procedure

- When you set up the plug-in for external share in IBM Content Navigator in a container environment (instead of an on-premises installation), use a URL value instead of a local path to point to the plug-in.

  In the wizard for adding the Share plug-in, for the JAR file path field, enter the following value: `http://`*`server`*`:`*`extshare-port`*`/contentapi/plugins/sharePlugin.jar`

  Where:

  – *server* is the value for the IBM Cloud Private server where your containers are deployed.

  – *extshare-port* is the port for your external share container.

  Or

  If using Ingress, enter the following value: `https://`*`<ingress-host-name>`*`/contentapi/plugins/sharePlugin.jar`, for example, `https://hostname.ingress.ibm.com/contentapi/plugins/sharePlugin.jar`

What to do next

Continue with the steps in Configuring external share in the IBM Content Navigator Knowledge Center.

Note: If you are using Ingress, any URL values that you specify in the configuration steps start with this pattern:

https:<ingress-host-name>

Chapter 9. (V5.5.4 or later) Optional: Configuring the Content Services GraphQL API

Content Services GraphQL API is an intuitive API that enables the caller to create, retrieve, update, or delete resources. The API is ideal for web and mobile application development because it supports retrieving exactly the data you need with a single call. You can make the API available for use by deploying the Content Services GraphQL API container.

Before you begin

Before you configure your container environment and deploy the Content Services GraphQL API container, deploy and configure the Content Platform Engine V5.5.4 or later container. Verify that the container deployment is working as intended.

About this task

The Content Services GraphQL API provides a schema and an easy-to-understand query language system that simplifies application development for your Content Platform Engine. The API schema definition of types and fields matches Content Engine Java API object model closely, with necessary and desirable extensions for natural GraphQL developer consumption.

You can find more information about using the API in Content Services GraphQL development.

To make the API available, you prepare the container environment, configure authentication, and deploy the Content Services GraphQL API container.

Creating volumes and folders for the Content Services GraphQL API container

The Content Services GraphQL API container requires a set of 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 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.

The following steps illustrate example YAML file contents to create a persistent volume and persistent volume claim for the GraphQL API container deployment.

Procedure

To create volumes and folders for the Content Services GraphQL container:

1. Create the `pv-graphql-cfgstore.yaml` file for the graphql-cfgstore configuration persistent volume:

```
apiVersion: v1
      kind: PersistentVolume
      metadata:
           annotations:
           name: graphql-cfgstore
         spec:
           accessModes:
           - ReadWriteMany
           capacity:
             storage: 1Gi
           nfs:
             path: /graphql/configDropins/overrides
             server:
           persistentVolumeReclaimPolicy: Retain
```

2. Save the file to a path in the cluster, for example, `pv-graphql-cfgstore.yaml`.
3. Create the persistent volume by running the following command:

```
kubectl apply -f pv-graphql-cfgstore.yaml
```

4. Create the `pvc-graphql-cfgstore.yml` file for the configuration persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: graphql-cfgstore
  namespace:
spec:
  accessModes:
  - ReadWriteMany
   resources:
     requests:
       storage: 1Gi
   volumeName: graphql-cfgstore-pv
```

5. Create the persistent volume claim by running the following command:

```
kubectl apply -f pvc-graphql-cfgstore.yaml
```

6. Create the `pv-graphql-logstore.yaml` file for the logging persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
  name: graphql-logstore-pv
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  nfs:
    path: /graphql/logs
    server:
  persistentVolumeReclaimPolicy: Retain
```

7. Create the persistent volume by running the following command:

```
kubectl apply -f pv-graphql-logstore.yaml
```

8. Create the `pvc-graphql-logstore.yaml` file for the logging persistent volume claim:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: graphql-logstore
  namespace:
spec:
  accessModes:
  - ReadWriteMany
   resources:
     requests:
       storage: 1Gi
   volumeName: graphql-logstore-pv
```

9. Create the persistent volume claim by running the following command:

```
kubectl apply -f pvc-graphql-logstore.yaml
```

10. Set the following permission on the mount path folders:

```
chown -Rf 50001:0 /graphql
chmod -Rf g=u /graphql
```

Deploying the Content Services GraphQL API container

Deploy the GraphQL API container on IBM Cloud Private or on an instance of certified Kubernetes where your Content Platform Engine and IBM Content Navigator containers are deployed.

Before you begin

Before you deploy this container release, ensure that you have configured the following containers in your container environment:

- Content Platform Engine V5.5.4 or later
- IBM Content Search Services V5.5.4 or later (optional if you want to enable CBR)

Make sure that your updated Content Platform Engine and IBM Content Search Services are configured before you deploy this GraphQL API container and complete the configuration of the GraphQL API feature.

This procedure assumes that you have completed the previous preparation tasks for deploying the Content Services GraphQL API container, including creating volumes and folder.

About this task

Instructions and sample files for deploying the containers on Kubernetes are provided in GitHub. The following deployment options are available:

Table 22. Options for deployment with the Cloud Pak operator

Platform	**Where to go**
Managed Red Hat OpenShift on IBM Cloud Public	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/roks/README.md
Red Hat OpenShift	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/ocp/README.md
Certified Kubernetes	https://github.com/ibm-ecm/container-samples/tree/5.5.4/operator/platform/k8s/README.md

Configuring cross origin resource sharing (CORS)

Cross origin resource sharing (CORS) allows requests from a different URL to be processed by the REST service that is running in the API container, and allows responses to be returned.

About this task

You enable the resource sharing by updating and adding the `CORS.xml` file to the `/configDropins/override` directory for the Content Services GraphQL API container.

Note: Rename the CORS.xml in the override directory to `zCORS.xml`. This ensures a file order that means this version of the cross origin resource sharing will override any other version.

Procedure

To configure the `zCORS.xml` file:

1. Copy the `zCORS.xml` sample file from the following Github location: https://github.com/ibm-ecm/container-samples/tree/5.5.4/ContentGraphQL/configDropins/overrides
2. Save the copy to the `/configDropins/override` directory for your Content Services GraphQL API container.
3. Edit the **`allowedOrigins`** parameter and enter the client domain names.

   The domain name consists of HTTP (non-SSL) or HTTPS (SSL), and the IBM Content Navigator host name and port that matches the requests coming from the client.

For example:

```
<server>

    <!-- https://www.ibm.com/support/knowledgecenter/en/SSEQTP_liberty/
com.ibm.websphere.liberty.autogen.nd.doc/ae/rwlp_config_cors.html -->

    <cors domain="/content-services-graphql"

        allowedOrigins="https://www.domain1.com:port, http://www.domain1.com:port, https://
www.domain2.com:port, http://www.domain2.com:port"

        allowedMethods="GET, POST"

        allowedHeaders="Connection,Pragma,Cache-Control,XSRFtoken,Origin,User-Agent,Content-
Type,Content-Length,Accept-Control-Request-Method,Accept-Control-Request-
Headers,Accept,Referer,Accept-Encoding,Accept-Language,DNT,Host,Content-Length,Cache-
control,Cookie"

       exposeHeaders="Content-Length,Content_Type,Content-Language,X-Powered-
By,Date,Allow,Transfer-Encoding,$WSEP,DNT,Access-Control-Allow-Credentials,Access-Control-
Allow-Headers,Access-Control-Allow-Max-Age,Access-Control-Allow-Methods,Access-Control-Allow-
Origin,Access-Control-Expose-Headers,Connection,Cache-control,Cookie"

        allowCredentials="true"

        maxAge="3600" />

</server>
```

You can add both HTTP and HTTPS values for the **allowedOrigins** value. Use a comma to separate multiple values for the entry.

Trailing slashes (/) are not allowed.

Important: Do not modify any parameter values other than the **allowedOrigins** value.

Configuring advanced authentication

Basic authentication, configured by using LDAP, includes browser authentication and API authentication. These methods might be sufficient in a test or development environment. However, if you plan to use the API in a production environment, it is recommended to configure advanced authentication by using OAuth 2.0 or OpenID Connect authentication.

Before you begin

The basic authentication setup and steps for preparing your environment, including the shared LDAP configuration, are prerequisites for configuring advanced authentication.

You also need to set up an identity provider that supports OAuth2.0 or OpenID Connect before you configure authentication.

About this task

Any identity providers that support OAuth2.0 or OpenID Connect can be used for this advanced authentication. A prerequisite for this configuration is that user names in the OAuth 2.0 / OpenID Connect Identity Provider must be identical to the user names in the LDAP server that Content Platform Engine servers are configured to use. The following steps demonstrate how OAuth 2.0 or OpenID Connect is configured using a User Management Service (UMS) server as the identity provider. Configuration options might be different if you are using a different type of identity provider.

Procedure

To configure advanced authentication:

1. Make sure that your identity provider supports JWK token validation using the RS256 signature algorithm.

   For UMS, for example, ensure that the following properties are set in the openidConnectProvider configuration:

- jwkEndpointUrl="https://UMS_HOST/oidc/endpoint/ums/jwk"
- signatureAlgorithm="RS256"

2. Register the Content Services GraphQL API with the identity provider.

   All identity providers (IdP) supporting OAuth 2.0 and OpenID Connect authentication have some registration mechanism to identify the client application to the IdP. At a minimum a client id, client secret, and redirect url(s) to the client application are required by the OAuth 2.0 and OpenID Connect specifications.

   You can post a JSON like the following example to a UMS registration endpoint to register a content services client. A description of each of these parameters can be found in the following location: https://www.ibm.com/support/knowledgecenter/SSEQTP_liberty/com.ibm.websphere.wlp.doc/ae/twlp_client_registration.html.

   JSON example:

```
POST https://<UMS host>:<UMS port>/oidc/endpoint/ums/registration
 {
            "token_endpoint_auth_method": "client_secret_basic",
            "scope": "openid profile",
            "grant_types": [
                "authorization_code"
            ],
            "response_types": [
                "code"
            ],
            "application_type": "web",
            "preauthorized_scope": "openid profile",
            "introspect_tokens": true,
            "resource_ids": [],
            "proofKeyForCodeExchange": false,
            "publicClient": false,
            "appPasswordAllowed": false,
            "appTokenAllowed": false,
            "hash_itr": 0,
            "hash_len": 0,
            "client_id": "graphql",
            "client_name": "graphql",
            "redirect_uris": [
                "https://<ContentGraphQL_hostName>/oidcclient/redirect/graphql"
            ],
            "allow_regexp_redirects": false,
            "client_secret": "************"
        }
```

   Notes:

   - Multiple redirect_uris can be specified if there are multiple Content Services GraphQL servers. Update both the trusted_uri_prefixes and redirect_uris sections to be consistent, for example:

```
"redirect_uris": [
       "https://servername:port/oidcclient/redirect/ContentServicesUms",
       "https://servername2:port2/oidcclient/redirect/ContentServicesUms",
        "regexp:https://servername3:!d*/oidcclient/redirect/ContentServicesUms"
   ],
```

   - If allow_regexp_redirects is set to true, you can use regular expressions in the redirect_uris instead of listing all of the Content Services hosts.

3. Import the identity provider into the Content Services GraphQL API trust store.

   To verify that the identity provider validation endpoint URI is trusted, the SSL certificate associated to that URI must be imported into the Content Services trust store.

   a) For a UMS server, use a command similar to the following to export its default SSL certificate into a file that will be imported in to the Content Services trust store:

```
keytool -exportcert -keystore your_keystore_filename -storepass
your_ums_keystore_password -alias default -rfc -file umscert.pem
```

b) Create a new trust store in the `${server-config-dir}/configDropins/overrides` directory and import the UMS certificate to the Content Services server by using a command similar to the following example:

```
keytool -import -file umscert.pem -alias ums -keystore graphqlTrustStore.p12 -storetype
pkcs12 -storepass your_password
```

c) Use the following list command to make sure that all keys are imported correctly:

```
keytool -list -keystore your_keystore_filename -storetype your_keystore_type -storepass
your_password
```

4. Define the identity provider client configuration for your Content Services GraphQL API environment:

a) Create a file called `zoidc.xml`, using the following example contents, and save it to the `${server-config-dir}/configDropins/overrides` folder for your Content Services Liberty server:

```
<?xml version='1.0' encoding='UTF-8'?>
<server>
    <featureManager>
        <feature>openidConnectClient-1.0</feature>
    </featureManager>

    <openidConnectClient id="graphql"
                         mapIdentityToRegistryUser="true"
                         scope="openid profile email"
                         responseType="code"
                         clientId="graphql"
                         clientSecret=""
                         issuerIdentifier="https://<UMS host>:<UMS port>/oidc/endpoint/
ums"
                         audiences="contentServicesUms"
                         trustStoreRef="graphqlTrustStore"
                         trustAliasName="ums"
                         validationMethod="introspect"
                         inboundPropagation="supported"
                         signatureAlgorithm="RS256"
                         jwkEndpointUrl="https://<UMS host>:<UMS port>/oidc/endpoint/ums/
jwk"
                         validationEndpointUrl="https://<UMS host>:<UMS port>/oidc/
endpoint/ums/introspect"
                         authorizationEndpointUrl="https://<UMS host>:<UMS port>/oidc/
endpoint/ums/authorize"
                         tokenEndpointUrl="https://<UMS host>:<UMS port>/oidc/
endpoint/ums/token">
    </openidConnectClient>

    <keyStore id="graphqlTrustStore" location="/opt/ibm/wlp/usr/servers/defaultServer/
configDropins/overrides/graphqlTrustStore.p12" type="PKCS12" password="changeit" />


</server>
```

Consider the following important points about this configuration:

- The values for **`clientId`** and **`clientSecret`** must match the **`client_id`** and **`client_secret`** values that you used when you registered the client with the identity provider.
- The Content Services server with this configuration must run on a host matching one of the redirect_uris that you specified when you registered the client with the identity provider.
- The value for id, for example, "graphql", must match the target of a redirect_uri in the registration with the identity provider. For example, https://servername:port/oidcclient/redirect/graphql.
- The value for **`issuerIdentifier`** must match how the identity provider, for example, UMS, sees its URI. If UMS is running in a Docker container under port 9443, but due to port mapping the UMS server is actually accessed externally through a different port number, the **`issuerIdentifier`** port must be specified as 9443. This value reflects how the Liberty server inside the Docker container sees its URI.
- The value for audiences must include your **`clientId`** plus the **`clientId`** of other applications that might invoke Content Services with an OpenID Connect token. Depending on your identity

provider, additional configuration might be required to support this scenario. For more details, refer to https://www.ibm.com/support/knowledgecenter/SSD28V_9.0.0/com.ibm.websphere.wlp.core.doc/ae/twlp_oidc_rsa_sha.html.

The identity provider uses the **`clientId`** setting to verify that the request is coming from a valid location. The Content services server uses the audiences setting to verify that the received authentication token was meant for it and not for some other application.

b) Update the keyStore stanza for the graphqlTrustStore into which you imported the UMS certificate.

5. An alternative for creating a separate trust store for your identity provider SSL certificates, you can override the Liberty server's default trust store and import the SSL certificates there.

a) Create a file called `z-ibm_custom-ssl.xml` with the following contents:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <server>
        <sslDefault sslRef="cssSSLSettings"/>

        <!-- SSL keystore and truststore configuration for BAN -->
        <ssl id="cssSSLSettings""
            keyStoreRef="customKeyStore"
            trustStoreRef="customTrustStoreOAUTH"
            clientAuthenticationSupported="false"
            sslProtocol="TLSv1.2"
            securityLevel="CUSTOM"
            enabledCiphers="TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256"
            />
        <keyStore id="customKeyStore"
            location="ibm_customBANKeyStore.p12"
            type="PKCS12" password="{xor}PDc+MTg6Nis=" />
        <keyStore id="customTrustStore"
            location="ibm_customBANTrustStore.p12"
            type="PKCS12" password="{xor}PDc+MTg6Nis=" />
        <keyStore id="customTrustStoreOAUTH"
            location="/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/
oauth.p12"
            type="PKCS12" password="{xor}PDc+MTg6Nis=" />

    </server>
```

b) Import your identity provider's SSL certificates into the `/opt/ibm/wlp/usr/servers/defaultServer/configDropins/overrides/oauth.p12` trust store by using a command like the following example:

```
keytool -import -file umscert.pem -alias ums -keystore oauth.p12 -storetype pkcs12 -
storepass your_password
```

c) Because you are overriding the default Liberty trust store, you can remove these two unnecessary attributes in the openidConnectClient configuration and the graphQL specific keystore definition:

```
    trustStoreRef="graphqlTrustStore"
    trustAliasName="ums"
```

In this scenario, then, your updated `zoidc.xml` file would look like the following example:

```
    <?xml version='1.0' encoding='UTF-8'?>
    <server>
       <featureManager>
           <feature>openidConnectClient-1.0</feature>
       </featureManager>

       <openidConnectClient id="ContentServicesUms"
                            mapIdentityToRegistryUser="true"
                            scope="openid profile email"
                            responseType="code"
                            clientId="contentServicesUms"
                            clientSecret=""
                            issuerIdentifier="https://<UMS host>:<UMS port>/oidc/endpoint/
ums"
                            audiences="contentServicesUms"
                            validationMethod="introspect"
```

```
                              inboundPropagation="supported"
                              signatureAlgorithm="RS256"
                              jwkEndpointUrl="https://<UMS host>:<UMS port>/oidc/
endpoint/ums/jwk"
                              validationEndpointUrl="https://<UMS host>:<UMS port>/oidc/
endpoint/ums/introspect"
                              authorizationEndpointUrl="https://<UMS host>:<UMS port>/oidc/
endpoint/ums/authorize"
                              tokenEndpointUrl="https://<UMS host>:<UMS port>/oidc/
endpoint/ums/token">
       </openidConnectClient>

    </server>
```

Enabling the Content Services GraphQL API trace log

You can set the logging levels and the output file for tracing operations for GraphQL API on the Liberty profile application server.

About this task

When you try to diagnose problems for GraphQL API, it is important to be able to see the log messages. To print readable log messages in log files, you must specify the applicable settings.

Procedure

To enable logging for GraphQL API:

- In the GraphQL configuration store `configDropins/overrides` directory, create a file called `liberty-logging.xml`, with the following contents:

  ```
      <server>
          <logging
  traceSpecification="com.ibm.ecm.content.graphql.*=finest:com.ibm.ecm.content.graphql.resolver
  s.*=finest"
          traceFileName="trace.log"
          maxFileSize="20"
          maxFiles="10"
          traceFormat="BASIC" />
      </server>
  ```

  Notes:

  The *=finest setting is equivalent to ALL.

  The `trace.log` file is only created if additional or detailed trace is enabled.

  If the logging level is info/audit/warning/severe/fatal, information is logged into `message.log` and `trace.log` is not created.

  If the logging level is config/detail/fine/finer/finest/all, information is logged into `trace.log` and the `trace.log` file is created.

  For more details, see Liberty profile: Logging and Trace.

Verifying the Content Services GraphQL API deployment

After you have deployed and configured the Content Services GraphQL API, you can verify the deployment and test your connection to Content Platform Engine

Procedure

To validate the GraphQL deployment:

- Access the GraphQL ping page with basic authentication, using a login URL like the following example:

  https://graphql-<project_name>.<Infrastructure_node_IP_or_hostname>.nip.io/content-services-graphql/ping

The product information displays in your browser:

```
{
"Build-Date" : "October 31, 2019 at 16:50",
"Implementation-Title" : "IBM FileNet Content Services GraphQL API - content-graphql-api",
"Implementation-Version" : "20191031-1650-415-Administrator",
"Product-Version" : "5.5.4",
"Build-Number" : "415"
}
```

- If you have GraphiQL enabled, you can validate the API connection to your Content Platform Engine.

  If you have basic authentication configured, you can access the GraphQL API with your basic authentication, using a login URL like the following example:

  https://graphql-<project_name>.<Infrastructure_node_ip_or_hostname>.nip.io/content-services-graphql

  You can use a query like the following example to test the connection to Content Platform Engine:

```
{
  _apiInfo(repositoryIdentifier: "ObjectStore1") {
    buildDate
    buildNumber
    implementationVersion
    implementationTitle
    productVersion
    cpeInfo {
      cpeURL
      cpeUser
      repositoryName
    }
  }
}
```

  The value for the repositoryIdentifier is the Content Platform Engine object store name (symbolic name) or id (GUID). A successful connection provides a return like the following example:

```
{
"data": {
"_apiInfo": {
"buildDate": " October 31, 2019 at 16:50 ",
"buildNumber": "415",
"implementationVersion": "20190529-0058-198-user123",
"implementationTitle": "IBM ECM Content Services GraphQL API - content-graphql-api",
"productVersion": "5.5.4",
"cpeInfo": {
"cpeURL": "http://cpe_host:port/wsi/FNCEWS40MTOM/",
"cpeUser": "CN=Admin,OU=Shared,OU=Engineering,OU=Company123,DC=adam2016,DC=com",
"repositoryName": "Object Store 1"
}
}
}
```

Chapter 10. 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 the following information:

- Administering Content Platform Engine
- Administering IBM Content Navigator components
- Administering IBM ECM CMIS

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 in the IBM Cloud Private console and from the Kubectl command line.

Important: Before you stop your containers, make sure to quiesce all processes that are running against the FileNetEngine application.

Procedure

- Scale your deployment in the IBM Cloud Private console:
  a) Navigate to **Workloads** > **Deployments** and locate the deployment you want to scale.
  b) From the **Action** menu for your deployment, choose **Scale**, and modify the number of instances.
     Setting the number to zero (0) stops all containers, and setting the number back to a non-zero value starts new instances.
- 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 the IBM Cloud Private 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.

About this task

If you deployed on IBM Cloud Private, you can use the console to monitor logs and events. For an externalized view of events and logs, you can use the Bluemix® dashboard or the dashboard for your ELK/GGC stack.

Procedure

- To monitor your logs and events in IBM Cloud Private, log into the console, and navigate to one of the following choices:
  - **Workloads** > **Deployments** > **Product_name** > **Events**
  - **Workloads** > **Deployments** > **Product_name** > **Logs**
- To monitor your logs and events in Bluemix Logmet, you can find your metrics data by creating or accessing your Bluemix dashboard.

  To access the monitoring dashboard, see the following information: https://console.bluemix.net/docs/services/cloud
- To monitor your logs and events in an ELK stack or GGB stack, you can find your data in the dashboards provided by these tools.

  To access the monitoring dashboard, follow the instructions provided by the ELK stack and GGB stack web information.

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 Compute Resources for Containers

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 IBM Cloud Private" on page 48.

  For the rest of your environment, the backup requirements are the same as for an on-premises installation. See the following information for details of what to back up: Backing up the data in your FileNet P8 domain.

  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. For more details, see the following information: Recovering the data in your FileNet P8 domain

  You must also recover the external container mounts for deployed containers as documented in the following topic: "Creating volumes and folders for deployment on IBM Cloud Private" on page 48.

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

Creating index areas for Content Search Services containers

You must complete several steps to make an index area available for use with a Content Search Services instance that is deployed as a container.

About this task

For an overview of index areas please see the topic Creating an index area.

After you prepare the volumes and folders, you use the Administration Console for Content Platform Engine to create the index area.

Procedure

To create an index area for Content Search Services on containers:

1. Prepare the file system using the persistent volume that was provided when the Content Search Services containers were deployed.

   See Preparing storage for FileNet P8 file storage areas and index areas for details on requirements for the storage.
2. Create a folder to represent the index area under the location used for the persistent volume claim given to the Content Search Services deployment.

   For example if the Content Search Services full text indexes volume claim is to `/indexes/FMdomain_Marketing`, you might create a folder for the index area at `/data/FMdomain_Marketing/indexArea3`.
3. Use the `chown` command to ensure that the folder representing the index area will be accessible to the Content Search Services containers.

   For example:

   (V5.5.2)

   ```
   chown -Rf 50001:50000 /indexes/FMdomain_Marketing/indexArea3
   ```

   (V5.5.1)

   ```
   chown -Rf 501:500 /indexes/FMdomain_Marketing/indexArea3
   ```
4. Determine the **Root Directory Path** value for the index area, based on the mountPath from inside the container.

   The path is `/opt/ibm/indexareas`. Using the example from the previous step, your **Root Directory Path** would be `/opt/ibm/indexareas/indexArea3`.
5. Create the index area:

   a) Log in to the Administration Console for Content Platform Engine and navigate to **Domain** > **Object stores** > ***object_store*** > **Administrative** > **Index areas**.

   b) Choose **New**.

   c) Complete the wizard to create the index area.

Creating advanced storage devices for Content Platform Engine containers

You configure security and prepare the environment before using the Administration Console for Content Platform Engine to create the advanced storage device.

About this task

For an overview of advanced storage devices, see the topic Advanced storage devices.

Procedure

1. If the advanced storage device will be configured for HTTPS certificate validation, you must obtain a client SSL certificate and import it into the truststore for the Content Platform Engine containers.
   a) For information about the client SSL certificate, see Obtaining a client SSL certificate.
   b) For instructions for how to create a custom keystore for the Content Platform Engine containers, see step 2 in Enabling custom SSL certificates for Content Search Services.
2. Prepare the environment according to which type of advanced storage device you plan to use:
   - File System Storage device
     a. Prepare the file system using the persistent volume that was provided when the Content Platform Engine containers were deployed. See Preparing storage for FileNet P8 file storage areas and index areas for details on requirements for the storage.
     b. Create a folder to represent the advanced storage device under the location used for the persistent volume claim given to the Content Platform Engine deployment. For example, if the Content Platform Engine filestore volume claim is to `/data/FMdomain_Marketing`, you might create a folder for the file system advanced storage device at `/data/FMdomain_Marketing/ASAD2`.
     c. Use the `chown` command to ensure the folder representing the advanced storage device are accessible to the Content Platform Engine containers. For example:

        (V5.5.2)

        ```
        chown –Rf 50001:50000 /data/FMdomain_Marketing/ASAD2
        ```

        (V5.5.1)

        ```
        chown –Rf 501:500 /data/FMdomain_Marketing/ASAD2
        ```

     d. Determine the value for the Root Directory Path based on the mountPath from inside the container, `/opt/ibm/asa`. Using the example from the earlier step, your **Root Directory Path** would be `/opt/ibm/asa/ASAD2`.
   - S3 Storage device
     a. Refer to Creating an S3 storage device for detailed information.
     b. To supply JVM arguments, add them into a file named `jvm.options` and put that file in the persistent volume location provided for the Content Platform Engine Liberty configuration data directory, for example, `/cpecfgstore/cpe/configDropins/overrides`. JVM arguments referenced are prefixed with –Dcom.filenet. when placed into the `jvm.options` file. For example, the option **`Advanced.S3.DeleteSpecificVersion`**, if set to true, would be a line in a file named `jvm.options` appearing as:

        ```
        -Dcom.filenet.Advanced.S3.DeleteSpecificVersion=true
        ```

3. Create the advanced storage device:
   a) Log in to the Administration Console for Content Platform Engine and navigate to **Domain** > **Object stores** > ***object_store*** > **Administrative** > **Storage** > **Advanced storage** > **Advanced storage devices**.
   b) Choose the S3 storage device type and click **New**.

c) Complete the wizard to create the advanced storage device.

(V5.5.2 and later) Optional: Enabling EMC Elastic Cloud Storage

You perform a simple additional configuration step to use EMC Elastic Cloud storage with your content services container environment.

About this task

You add the ECS Pool Entry Authorization (PEA) file to your container environment to enable communication with the EMC Elastic Cloud Storage device. You can perform this enablement step before or after you deploy the containers. The file must be present at runtime.

This additional step enables EMC Elastic Cloud Storage to work in a container environment. To use EMC Elastic Cloud storage as your fixed content device, you must also complete the typical configuration for the device in the Administration Console for Content Platform Engine.

For more information about EMC Elastic Cloud Storage, see the topic EMC Elastic Cloud Storage fixed content device.

Procedure

To enable EMC Elastic Cloud Storage:

1. Obtain the PEA file from the EMC Elastic Cloud Storage server.

   You can generate the Pool Entry Authorization (PEA) file from the Elastic Cloud Storage (ECS) administration console.
2. Copy the PEA file to the `https://www.ibm.com/support/knowledgecenter/en/SSNW2F_5.5.0/configDropins/overrides` directory for the Content Platform Engine container.

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.

Chapter 11. Troubleshooting your container environment

You can use the IBM Cloud Private administration console or the command line tools to check and correct the health of your container environment.

About this task

The troubleshooting steps assume that you have access to the IBM Cloud Private administration console, or that you have the Kubectl command line tool installed and configured.

Procedure

- Get IBM Cloud Private nodes information:

  IBM Cloud Private console
  Navigate to **Menu** > **Platform** > **Nodes**.

  Command line:
  Use the following command:

  ```
  kubectl get nodes -n namespace -o wide
  ```

- Review the Persistent Volume and Persistent Volume Claim information and make sure they match with your deployment YAML files:

  IBM Cloud Private console
  Navigate to **Menu** > **Platform** > **Storage** > **PersistentVolumes**.

  Navigate to **Menu** > **Platform** > **Storage** > **PersistentVolumeClaims**.

  Command line:
  Use the following command:

  ```
  -kubectl describe pv pv name -n namespace
  ```

- Get the status of the content services containers deployment job:

  IBM Cloud Private console
  Navigate to **Menu** > **Platform** > **Storage** > **PersistentVolumes**.

  Navigate to **Menu** > **Workload** > **Deployments** > ***Deployment Name*** > **Events**.

  Command line:
  Use the following commands:

  ```
  kubectl -n <namespace> rollout status deployment/deployment name
  ```

  ```
  kubectl describe deployment deployment name -n namespace
  ```

- Get information about the pods after the content services containers have been deployed:

  IBM Cloud Private console
  Navigate to **Menu** > **Workload** > **Deployment** > **Deployment Name**. Scroll down to find the Pods section, and click the name of the pod that you want to check.

  Command line:
  Use the following command:

  ```
  --kubectl get pods -n namespace -o wide
  ```

- Retrieve the logs for a particular pod.

Use the following command:

```
-kubectl logs -f pod name -c deployment name -n namespace
```

- Get details about a particular pod.

  Use the following command:

```
-kubectl describe pods pod name -n namespace
```

Re-running a failed container deployment job

If a container deployment fails, you can investigate the cause and then re-run the deployment from the configuration tool or from the command line.

About this task

If the deployment of a container fails, first investigate and address the cause for the failure. Check the `ibacc.log` file for details of the deployment process. After that, you can re-run the failed job from the IBM Business Automation Configuration Container tool if you originally deployed the container from the configuration tool user interface. You can also re-run failed job from Kubectl command line whether you originally deployed from the configuration tool web UI or from the configuration tool command line.

Procedure

- Re-run a deployment with the configuration tool user interface.

  In the **Deployment progress** page of the configuration tool, click the **Action** drop-down list, and click **Re-run item**.
- Re-run a deployment with the configuration tool command line:
  a) Copy the generated YAML files for the job containers from the persistent volume for the configuration tool, by default `ibacc-cfg-pvc`, to your machine.

     Your machine must be able to access IBM Cloud Private and must have Kubectl installed.

     You might need to go one level up from the persistent volume, and navigate to a directory such as ecm_staging_deploy/cfg to find the YAML files. The generated files for the job containers are usually named like the following examples. Note that depending on your selection of containers, you might not see all of the files:

     – IBACC-ecm-provision-icp-job.yml

     – IBACC-init-deploy.yml

     – IBACC-verify-deploy.yml

  b) Run the Kubectl command to deploy the container:

     For example, to redeploy the provision job, run the following command:

```
Kubectl apply –f IBACC-ecm-provision-icp-job.yml
```

  c) When the deployment job completes, delete the pod for the job container manually to ensure that there is no name conflict later:

```
Kubectl get pod –a –n <namespace>
Kubectl delete pod <job pod name> -n <namespace>
```

  d) When the job completes successfully, you can run additional deployments with the `Kubectl apply -f` command.
- If the initialize and verify steps failed, you can manually perform the initialization.

  You can fix the environment by deleting the created object store, cleaning the associated tablespace, and recreating the object stores manually through the Administration Console for Content Platform Engine.

For more information about initialization, see the following topic: "Initializing and verifying your content services environment" on page 112.

Provisioning fails because of a persistent volume claim name

The deployment of a container fails if a persistent volume claim name contains a space.

About this task

If a persistent volume claim name that contains a space is entered into the IBM Business Automation Configuration Container tool, deployment fails immediately. For example, this situation might occur when a persistent volume claim name contains a space that was added accidentally at the end of the name.

Procedure

1. Delete the `deployed.ok` file inside the `ecm-staging-deploy` folder in the management node, for example, `/mnt/nfsshare/ibacc/ecm-staging-deploy/cfg/ecm_staging_deploy/cfg/deployed.ok`.
2. Re-run the IBM Business Automation Configuration Container tool and check each persistent volume claim value to ensure that no extra space is included.

Viewing container logs

You can obtain information about the operation of your container deployments by viewing the log files for each container.

About this task

You use the Kubectl command line tool to get details about the log, and display the log information.

Viewing the IBM Business Automation Configuration Container logs

Logs are stored in the defined persistent volume location.

Procedure

- View the logs from defined shared Kubernetes configuration persistent volume location.

  Change to the location of the corresponding persistent volume claim for the log that you want to view:

  – For the IBM Business Automation Configuration Container Master:

  ```
  /ibacc/ecm-staging-deploy/cfg/logs/ibacc-master
  ```

  – For IBM Business Automation Configuration Container - IBM Cloud Private:

  ```
  /ibacc/ecm-staging-deploy/cfg/logs/ibacc-ICp
  ```

  – For IBM Business Automation Configuration Container Initialization:

  ```
  /ibacc/ecm-staging-deploy/cfg/logs/ibacc-init
  ```

  – For IBM Business Automation Configuration Container Verification:

  ```
  /ibacc/ecm-staging-deploy/cfg/logs/ibacc-verify
  ```

- Alternatively, you can view the logs by accessing the corresponding job container pod:

  – For the IBM Business Automation Configuration Container Master:

  ```
  kubectl get deployment -a -n $NS
  kubectl get pod -a -n $NS
  kubectl logs $(kubectl get pod -a -n $NS | grep ibacc-master
   | head -1 | awk '{print $1}') -f -n $NS
  ```

- For IBM Business Automation Configuration Container - IBM Cloud Private:

  ```
  kubectl logs ecm-cfg-icp-container
  ```

- For IBM Business Automation Configuration Container Initialization:

  ```
  kubectl logs ecm-cfg-init-container
  ```

- For IBM Business Automation Configuration Container Verification:

  ```
  kubectl logs ecm-cfg-verify-container
  ```

Viewing the Content Platform Engine container logs

Logs are stored in the defined persistent volume location.

Procedure

- View the logs from defined shared Kubernetes configuration persistent volume location.
  a) Get the corresponding application pod name:

  ```
  kubectl get pod | grep cpe
  ```

  b) Change to the location of the corresponding persistent volume claim, for example, cpe-icp-logstore, for the log that you want to view:
  - For the `messages.log` and `console.log`:

    ```
    /cpelogstore/cpe/logs/pod-name
    ```

  - For the runtime logs, such as `p8_server_error.log` and `p8_server_trace.log`:

    ```
    /cpefnlogstore/FileNet/pod-name
    ```

- Alternatively, you can view the logs by accessing the corresponding job container pod:
  a) Get the corresponding application pod name:

  ```
  kubectl get pod | grep cpe
  ```

  b) Access the application pod:

  ```
  kubectl exec –it pod-name bash
  ```

  Then change directories to view the logs:
  - For the `messages.log` and `console.log`:

    ```
    logs/pod-name
    ```

  - For the runtime logs, such as `p8_server_error.log` and `p8_server_trace.log`:

    ```
    FileNet/pod-name
    ```

Viewing the IBM Content Navigator container logs

Logs are stored in the defined persistent volume location.

Procedure

- View the logs from defined shared Kubernetes configuration persistent volume location.
  a) Get the corresponding application pod name:

  ```
  kubectl get pod | grep icn
  ```

b) Change to the location of the corresponding persistent volume claim, for example, icn-icp-logstore, for the log that you want to view:

   – For the `messages.log` and `console.log`:

   ```
   /icnlogstore/icn/logs/pod-name
   ```

- Alternatively, you can view the logs by accessing the corresponding job container pod:

  a) Get the corresponding application pod name:

  ```
  kubectl get pod | grep icn
  ```

  b) Access the application pod:

  ```
  kubectl exec –it pod-name bash
  ```

  Then change directories to view the logs:

  – For the `messages.log` and `console.log`:

  ```
  logs/pod-name
  ```

Viewing the Content Management Interoperability Services container logs

Logs are stored in the defined persistent volume location.

Procedure

- View the logs from defined shared Kubernetes configuration persistent volume location.

  a) Get the corresponding application pod name:

  ```
  kubectl get pod | grep cmis
  ```

  b) Change to the location of the corresponding persistent volume claim, for example, icn-cmis-logstore, for the log that you want to view:

  – For the `messages.log` and `console.log`:

  ```
  /cmislogstore/cmis/logs/pod-name
  ```

- Alternatively, you can view the logs by accessing the corresponding job container pod:

  a) Get the corresponding application pod name:

  ```
  kubectl get pod | grep cmis
  ```

  b) Access the application pod:

  ```
  kubectl exec –it pod-name bash
  ```

  Then change directories to view the logs:

  – For the `messages.log` and `console.log`:

  ```
  logs/pod-name
  ```

Viewing the Content Search Services container logs

Logs are stored in the defined persistent volume location.

Procedure

- View the logs from defined shared Kubernetes configuration persistent volume location.

  a) Get the corresponding application pod name:

```
kubectl get pod | grep css
```

b) Change to the location of the corresponding persistent volume claim, for example, icn-css-logstore, for the log that you want to view:

   – For the `messages.log` and `console.log`:

   ```
   /csslogstore/CSS_Server_log/csssearch_logs/pod-name
   ```

- Alternatively, you can view the logs by accessing the corresponding job container pod:

  a) Get the corresponding application pod name:

  ```
  kubectl get pod | grep css
  ```

  b) Access the application pod:

  ```
  kubectl exec –it pod-name bash
  ```

  Then change directories to view the logs:

  – For the `default0.log` and `trace0.log`:

  ```
  /opt/IBM/ContentSearchServices/CSS_Server/log/pod-name
  ```

Chapter 12. 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.

(V5.5.4 and later) 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 that are relevant for your overall FileNet Content Manager and IBM Content Navigator environment.

Table 23. Shared configuration parameters

Parameters	Description	Default Values
Deployment platform sc_deployment_platform	Enter your certified Kubernetes platform type	OCP
Deployment context sc_deployment_context	Do not change this default setting.	FNCM
Root ca secret root_ca_secret	If you provide your own root certificate, enter the value.	fncm-root-ca
Trusted certificate list 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.	[]

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 24. 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>

Table 24. LDAP configuration parameters (continued)

Parameters	Description	Default Values
LDAP port lc_ldap_port	The port number for the LDAP server that you are using.	389
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_att r	Provide the format of the group display name.	cn
Group membership search filter lc_ldap_group_membership_sea rch_filter	Filter for finding entries in the LDAP base DN (groups) subtree that match the group name.	(|(&(objectclass=groupofnames)(member={0}))(&(objectclass=groupofuniquena mes)(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.	

Table 24. LDAP configuration parameters (continued)

Parameters	**Description**	**Default Values**
(Active Directory) lc_ad_gc_port	Active Directory port.	
(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=groupofuniquenam es) (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=groupofuniquenam es) (objectclass=groupofurls)))
(External users) 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
(External users) LDAP server host name lc_ldap_server	The host name for the LDAP server that you are using for the environment.	<hostname>
(External users) LDAP port lc_ldap_port	The port number for the LDAP server that you are using.	389
(External users) 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
(External users) SSL enablement lc_ldap_ssl_enabled	Specify whether SSL is enabled.	false
(External users) SSL secret lc_ldap_ssl_secret_name	Provide the name of the SSL secret that you created.	
(External users) User name attribute lc_ldap_user_name_attribute	Provide the format of the user name.	*:cn

Table 24. LDAP configuration parameters (continued)

Parameters	Description	Default Values
(External users) User display name attribute lc_ldap_user_display_name_attr	Provide the format of the display name.	cn
(External users) 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
(External users) Group name lc_ldap_group_name_attribute	Provide the format of the group name.	*:cn
(External users) Group display name lc_ldap_group_display_name_attr	Provide the format of the group display name.	cn
(External users) 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})))
(External users) 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
(External users) Max search results lc_ldap_max_search_results	Maximum number of search results to return.	4500
(External users) (Active Directory) lc_ad_gc_host	Active Directory host.	
(External users) (Active Directory) lc_ad_gc_port	Active Directory port.	
(External users) (Active Directory) User filter lc_user_filter	Active Directory user filter.	(&(cn=%v)(objectclass=person))
(External users) (Active Directory) Group filter lc_group_filter	Active Directory group filter.	(&(cn=%v)(|(objectclass=groupofnames)(objectclass=groupofuniquenames)(objectclass=groupofurls)))

Table 24. LDAP configuration parameters (continued)		
Parameters	**Description**	**Default Values**
(External users) (IBM Security) User filter lc_user_filter	IBM Security user filter	(&(cn=%v)(objectclass=person))
(External users) (IBM Security) Group filter lc_group_filter	IBM Security group filter.	(&(cn=%v) (|(objectclass=groupofnames) (objectclass=groupofuniquenam es) (objectclass=groupofurls)))

Datasource parameters

Update the custom YAML file to provide the details that are relevant for your FileNet Content Manager and IBM Content Navigator datasource environment. The default parameter list assumes one Content Platform Engine Global Configuration database and two object store databases, and one IBM Content Navigator databases. Some parameters are specific to database vendors.

Table 25. Datasource configuration parameters		
Parameters	**Description**	**Default Values**
(GCD) Database type dc_database_type	Specify the type for your GCD database	db2
(GCD) Datasource name dc_common_gcd_datasource_na me	The JNDI name of the non-XA JDBC data source associated with the global configuration table space or database. The name must be unique.	FNGCDDS
(GCD) XA datasource name dc_common_gcd_xa_datasource _name	The JNDI name of the XA JDBC data source associated with the global configuration table space or database. The name must be unique.	FNGCDDSXA
(GCD) Database server host database_servername	The host name of the server where the database software is installed.	<hostname>
(GCD) Database name database_name	Provide the database name.	GCDDB
(GCD) Database port # database_port: ""	Provide the database port.	50000
(GCD) Database URL dc_oracle_gcd_jdbc_url	Oracle: Provide the URL for the database.	jdbc:oracle:thin:@// <hostname>:1521/orcl
(GCD) Standby server name dc_hadr_standby_servername	Enter the standby server name.	<hostname>

Table 25. Datasource configuration parameters (continued)

Parameters	Description	Default Values
(GCD) Standby server port dc_hadr_standby_port	Enter the standby server port.	50000
(GCD) Validation timeout dc_hadr_validation_timeout	Specify the validation timeout entry.	15
(GCD) 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.	15
(GCD) 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
(OS1) Database type dc_database_type	Specify the type for your Object Store database	db2
(OS1) Datasource name dc_common_os_datasource_name	The JNDI name of the non-XA JDBC data source associated with the object store table space or database. The name must be unique.	FNOS1DS
(OS1) XA datasource name dc_common_os_xa_datasource_name	The JNDI name of the XA JDBC data source associated with the object store table space or database. The name must be unique.	FNOS1DSXA
(OS1) Database server host database_servername	The host name of the server where the database software is installed.	<hostname>
(OS1) Database name database_name	Provide the database name.	OS1DB
(OS1) Database port database_port	Provide the database port.	50000
(OS1) Database URL dc_oracle_os_jdbc_url	Oracle: Provide the URL for the database.	jdbc:oracle:thin:@//<hostname>:1521/orcl
(OS1) Standby server name dc_hadr_standby_servername	Enter the standby server name.	<hostname>
(OS1) Standby server port dc_hadr_standby_port	Enter the standby server port.	50000

Table 25. Datasource configuration parameters (continued)

Parameters	Description	Default Values
(OS1) Validation timeout dc_hadr_validation_timeout	Specify the validation timeout entry.	15
(OS1) 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.	15
(OS1) 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
(OS2) Database type dc_database_type	Specify the type for your Object Store database	db2
(OS2) Datasource name dc_common_os_datasource_name	The JNDI name of the non-XA JDBC data source associated with the object store table space or database. The name must be unique.	FNOS2DS
(OS2) XA datasource name dc_common_os_xa_datasource_name	The JNDI name of the XA JDBC data source associated with the object store table space or database. The name must be unique.	FNOS2DSXA
(OS2) Database server host database_servername	The host name of the server where the database software is installed.	<hostname>
(OS2) Database name database_name	Provide the database name.	OS2DB
(OS2) Database port database_port	Provide the database port.	50000
(OS2) Database URL dc_oracle_os_jdbc_url	Oracle: Provide the URL for the database.	jdbc:oracle:thin:@//<hostname>:1521/orcl
(OS2) Standby server name dc_hadr_standby_servername	Enter the standby server name.	<hostname>
(OS2) Standby server port dc_hadr_standby_port	Enter the standby server port.	50000
(OS2) Validation timeout dc_hadr_validation_timeout	Specify the validation timeout entry.	15

Table 25. Datasource configuration parameters (continued)

Parameters	**Description**	**Default Values**
(OS2) Retry interval for client reroute dc_hadr_retry_interval_for_clien t_reroute	Specify the time in seconds between connection attempts made by the automatic client reroute if the primary connection to the server fails.	15
(OS2) 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
(ICN) Database type dc_database_type	Specify the type for your IBM Content Navigator database	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_na me	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>
(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_clien t_reroute	Specify the time in seconds between connection attempts made by the automatic client reroute if the primary connection to the server fails.	3

Table 25. Datasource configuration parameters (continued)

Parameters	Description	Default Values
(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

Monitoring parameters

Update the custom YAML file to provide the details that are relevant for your FileNet Content Manager and IBM Content Navigator logging and monitoring environment.

Table 26. Logging and monitoring configuration parameters

Parameters	Description	Default Values
Monitoring metrics writer mon_metrics_writer_option	Provide the monitoring metrics option.	4
Monitoring metrics endpoint mon_metrics_service_endpoint	Provide the monitoring metrics service endpoint using the format host:port.	<hostname>:2003
Monitoring metrics IBM Cloud group mon_bmx_group	IBM Cloud metrics group name ended with ".", which is used for metrics prefix.	ibm
Monitoring metrics scope ID mon_bmx_metrics_scope_id	GUID that identifies the space or organization domain where the metrics are stored.	1
Monitoring metrics API key mon_bmx_api_key	IBM Cloud API key, which is used for all the RESTful APIs.	testkey
Collection interval mon_ecm_metrics_collect_interval	Seconds between collecting metrics.	60
Flush interval mon_ecm_metrics_flush_interval	Seconds between flushing metrics.	60
Enable performance plug-in mon_enable_plugin_pch	Performance metrics for FNCM components. (Do not update)	true
Enable JMX performance plug-in mon_enable_plugin_mbean	Performance metrics for JMX. (Do not update)	true
Enable log parsing mon_log_parse	Keep default- true	true
Log shipper mon_log_shipper_option	Provide the setting to determine the log shipper	1

Table 26. Logging and monitoring configuration parameters (continued)

Parameters	Description	Default Values
Log service endpoint mon_log_service_endpoint	Logging service endpoint using format host1:port, host2:port, ..., hostN:port. The port for all hosts must be same.	<hostname>:5044
Log token mon_bmx_logs_logging_token	Token for the logging service.	testtoken
Log IBM Cloud space ID mon_bmx_space_id	GUID that identifies the space or organization domain where the metrics are stored.	1

Content Platform Engine parameters

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

Table 27. Configuration parameters

Parameters	Description	Default Values
Replica count replica_count	How many Content Platform Engine replicas to deploy.	1
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/cpe ga-554-p8cpe 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.	3072Mi
Auto Scale enabled	Specify whether to enable auto scaling.	false

Table 27. Configuration parameters (continued)

Parameters	**Description**	**Default Values**
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
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"	None
GCD JNDI name gcd_jndi_name	JNDI name for the Global Configuration Database.	FNGCDDS
GCD JNDI XA name gcd_jndixa_name	JNDI XA name for the Global Configuration Database	FNGCDDSXA

Table 27. Configuration parameters (continued)

Parameters	**Description**	**Default Values**
License Model license_model	Choose the licensing model. Required. The expected values are ICF.PVUNonProd, ICF.PVUProd, ICF.UVU, ICF.CU, FNCM.PVUNonProd, FNCM.PVUProd, FNCM.UVU, or FNCM.CU.	FNCM.CU
Accept license license	The value must be set to accept to deploy.	accept
Enable monitoring monitor_enabled	Specify whether to use the built-in monitoring capability.	
Enable logging logging_enabled	Specify whether to use the built-in logging capability.	
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_cpe_cfgstore)	The persistent volume claim for Content Platform Engine configuration.	cpe-cfgstore
Logs PVC name (existing _pvc_for_cpe_logstore)	The persistent volume claim for Content Platform Engine logs.	cpe-logstore
File store PVC name (existing _pvc_for_cpe_filestore)	The persistent volume claim for the Content Platform Engine files.	cpe-filestore
IBM Case Manager rules PVC name (existing _pvc_for_cpe_icmrulesstore)	The persistent volume claim for the IBM Case Manager rules.	cpe-icmrulesstore
Text extraction PVC name (existing _pvc_for_cpe_textextstore)	The persistent volume claim for text extraction	cpe-textextstore
Bootstrap PVC name for upgrade (existing _pvc_for_cpe_bootstrapstore)	The persistent volume claim for upgrade and startup.	cpe-bootstrapstore
FileNet logs PVC name (existing _pvc_for_cpe_fnlogstore)	The persistent volume claim for FileNet logs.	cpe-fnlogstore

Table 27. 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

Content Search Services parameters

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

Table 28. Content Search Services configuration parameters

Parameters	Description	Default Values
Replica count replica_count	How many replicas to deploy.	1
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/css ga-554-p8css 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

Table 28. Content Search Services configuration parameters (continued)

Parameters	**Description**	**Default Values**
Resource -> limits memory	Specify a memory limit for the container.	3072Mi
Maximum percentage jvm_max_heap_percentage	The maximum percentage of available memory to use.	33%
Accept license license	The value must be set to accept to deploy.	accept
Enable monitoring monitor_enabled	Specify whether to use the built-in monitoring capability.	
Enable logging logging_enabled	Specify whether to use the built-in logging capability.	
Enable graphite plugin collectd_enable_plugin_write_graphite	If you have a Graphite database for metrics, or if you use IBM Cloud monitoring, set to true.	false
Configuration overrides PVC name (existing _pvc_for_cpe_cfgstore)	The persistent volume claim for Content Platform Engine configuration.	cpe-cfgstore
Logs PVC name (existing _pvc_for_cpe_logstore)	The persistent volume claim for Content Platform Engine logs.	cpe-logstore
File store PVC name (existing _pvc_for_cpe_filestore)	The persistent volume claim for the Content Platform Engine files.	cpe-filestore
IBM Case Manager rules PVC name (existing _pvc_for_cpe_icmrulesstore)	The persistent volume claim for the IBM Case Manager rules.	cpe-icmrulesstore
Text extraction PVC name (existing _pvc_for_cpe_textextstore)	The persistent volume claim for text extraction	cpe-textextstore
Bootstrap PVC name for upgrade (existing _pvc_for_cpe_bootstrapstore)	The persistent volume claim for upgrade and startup.	cpe-bootstrapstore
FileNet logs PVC name (existing _pvc_for_cpe_fnlogstore)	The persistent volume claim for FileNet logs.	cpe-fnlogstore

Table 28. Content Search Services 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

Content Management Interoperability Services parameters

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

Table 29. CMIS configuration parameters

Parameters	Description	Default Values
Replica count replica_count	How many replicas to deploy.	1
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/cmis ga-304-cmis-if009 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.	256Mi
Resource -> limits cpu	Specify a CPU limit for the container.	1

Table 29. CMIS configuration parameters (continued)

Parameters	**Description**	**Default Values**
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_percentag e	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
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"	None

Table 29. CMIS configuration parameters (continued)

Parameters	**Description**	**Default Values**
checkout_copycontent	Determines whether the content-stream of the Private Working Copy should be copied from the Document that was checked out.	true
default_maxitems	The default value for the optional maxItems input argument on paging-related services.	25
cvl_cache	Determines whether ChoiceLists will be cached once for all users.	true
filter_hidden_properties	Determines whether hidden P8 domain properties should appear in CMIS type definitions and folder or document instance data.	true
querytime_limit	Timeout in seconds for the queries that specify timeout.	180
resumable_queries_forrest	If true, then a faster response time for REST next line. If false, the next link for REST will re-issue query.	true
escape_unsafe_string_characters	Specifies whether to escape characters that are not valid for XML unicode as specified by the XML 1.0 standard.	false
max_soap_size	Limits the maximum allowable Web Service SOAP message request size.	180
print_pull_stacktrace	Prints the full stack trace in the response.	false
folder_first_search	Configures the sequence in which CMIS tries to identify objects (folder or document first).	false
ignore_root_documents	To ignore reading or writing contents in root folder, set ignoreRootDocuments to true.	false
supporting_type_mutability	Determines whether to support type mutability.	false
License Model license_model	Choose the licensing model. Required. The expected values are ICF.PVUNonProd, ICF.PVUProd, ICF.UVU, ICF.CU, FNCM.PVUNonProd, FNCM.PVUProd, FNCM.UVU, or FNCM.CU.	FNCM.CU

Table 29. CMIS configuration parameters (continued)		
Parameters	**Description**	**Default Values**
Accept license license	The value must be set to accept to deploy.	accept
Enable monitoring monitor_enabled	Specify whether to use the built-in monitoring capability.	
Enable logging logging_enabled	Specify whether to use the built-in logging capability.	
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_cpe_cfgstore)	The persistent volume claim for Content Platform Engine configuration.	cpe-cfgstore
Logs PVC name (existing _pvc_for_cpe_logstore)	The persistent volume claim for Content Platform Engine logs.	cpe-logstore
File store PVC name (existing _pvc_for_cpe_filestore)	The persistent volume claim for the Content Platform Engine files.	cpe-filestore
IBM Case Manager rules PVC name (existing _pvc_for_cpe_icmrulesstore)	The persistent volume claim for the IBM Case Manager rules.	cpe-icmrulesstore
Text extraction PVC name (existing _pvc_for_cpe_textextstore)	The persistent volume claim for text extraction	cpe-textextstore
Bootstrap PVC name for upgrade (existing _pvc_for_cpe_bootstrapstore)	The persistent volume claim for upgrade and startup.	cpe-bootstrapstore
FileNet logs PVC name (existing _pvc_for_cpe_fnlogstore)	The persistent volume claim for FileNet logs.	cpe-fnlogstore
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

Table 29. CMIS configuration parameters (continued)

Parameters	Description	Default Values
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

Content Services GraphQL parameters

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

Table 30. GraphQL configuration parameters

Parameters	Description	Default Values
Replica count replica_count	How many replicas to deploy.	1
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/graphql ga-554-p8cgql 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

Table 30. GraphQL configuration parameters (continued)

Parameters	Description	Default Values
Auto Scale max_replicas	The upper limit for the number of pods that can be set by the autoscaler. Required.	1
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_percentag e	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
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"	None
License Model license_model	Choose the licensing model. Required. The expected values are ICF.PVUNonProd, ICF.PVUProd, ICF.UVU, ICF.CU, FNCM.PVUNonProd, FNCM.PVUProd, FNCM.UVU, or FNCM.CU.	FNCM.CU

Table 30. GraphQL configuration parameters (continued)

Parameters	**Description**	**Default Values**
Accept license license	The value must be set to accept to deploy.	accept
Enable GraphiQL	Specify whether to enable the GraphiQL UI. True is recommended for development environments. False is recommended for production environments.	false
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.	true
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_graphql_cfgstore)	The persistent volume claim for Content Platform Engine configuration.	graphql-cfgstore
Logs PVC name (existing _pvc_for_graphql_logstore)	The persistent volume claim for GraphQL logs.	graphql-logstore
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

External Share parameters

Update the custom YAML file to provide the details that are relevant to your External Share configuration and your decisions for the deployment of the container.

Table 31. External share configuration parameters

Parameters	Description	Default Values
Replica count replica_count	How many Content Platform Engine replicas to deploy.	1
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/extshare ga-307-gs Always
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

Table 31. External share configuration parameters (continued)		
Parameters	**Description**	**Default Values**
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.	40
Maximum percentage jvm_max_heap_percentage	The maximum percentage of available memory to use.	66
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"	None
License Model license_model	Choose the licensing model. Required. The expected values are ICF.PVUNonProd, ICF.PVUProd, ICF.UVU, ICF.CU, FNCM.PVUNonProd, FNCM.PVUProd, FNCM.UVU, or FNCM.CU.	FNCM.CU
Accept license license	The value must be set to accept to deploy.	accept
Database type es_dbtype	The database type for external share.	db2
JNDI datasource es_jindi_ds	The JNDI datasource for external share.	ECMClientDS
Schema es_schema	The schema for external share.	ICNDB
Tablespace es_ts	The table space for external share.	ICNDB

Table 31. External share configuration parameters (continued)

Parameters	Description	Default Values
Administrator es_admin	The administrator for external share.	ceadmin
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.	true
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_es_cfgstore)	The persistent volume claim for External Share configuration.	es-cfgstore
Logs PVC name (existing _pvc_for_es_logstore)	The persistent volume claim for External Share logs.	es-logstore
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

Task Manager parameters

Update the custom YAML file to provide the details that are relevant to your Task Manager and your decisions for the deployment of the container.

Table 32. Task Manager configuration parameters

Parameters	Description	Default Values
Replica count replica_count	How many Content Platform Engine replicas to deploy.	1

Table 32. Task Manager 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/taskmgr 3.0.7 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_percentag e	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
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%

Table 32. Task Manager configuration parameters (continued)

Parameters	Description	Default Values
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"	"-Dcom.ibm.ecm.task.StartUpListener.defaultLogLevel=FINE"
tm_dbtype		db2
tm_indi_ds		ECMClientDS
tm_schema		ICNDB
tm_ts		ICNDB
tm_admin		ceadmin
Accept license license	The value must be set to accept to deploy.	accept
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.	true
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_tm_cfgstore)	The persistent volume claim for Task Manager configuration.	tm-cfgstore
Logs PVC name (existing _pvc_for_tm_logstore)	The persistent volume claim for Task Manager logs.	tm-logstore
Plug in PVC name (existing_pvc_for_tm_pluginstore)	The persistent volume claim for the Task Manager plug in.	tm-pluginstore
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

Table 32. Task Manager configuration parameters (continued)

Parameters	Description	Default Values
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

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 33. 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.	1
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-307-icn Always
Logging for workloads (log) format:	The format for workload logging.	json
Resources -> requests cpu	Specifies a CPU request for the container.	500m

Table 33. Configuration parameters (continued)

Parameters	Description	Default Values
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_percentag e	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
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%

Table 33. Configuration parameters (continued)

Parameters	**Description**	**Default Values**
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"	None
IBM Content Navigator database type icn_db_type	Type of the IBM Content Navigator database.	db2
Navigator JNDI datasource name icn_jndids_name	Name for the Navigator JNDI datasource.	ECMClientDS
Schema icn_schema	Schema for IBM Content Navigator.	ICNDB
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

Table 33. Configuration parameters (continued)

Parameters	**Description**	**Default Values**
File store PVC name (existing _pvc_for_cpe_filestore)	The persistent volume claim for the Content Platform Engine files.	cpe-filestore
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.	inc-cw-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.	inc-asperastore
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

Initialization parameters

Update the custom YAML file to provide the details that are relevant for initializing your FileNet Content Manager and IBM Content Navigator components.

Table 34. Initialization configuration parameters

Parameters	**Description**	**Default Values**
domain_name	Provide a name for the domain.	P8DOMAIN
encryption_key	The encryption strength.	128
ic_ldap_admin_user_name	Administrator user.	ceadmin
ic_ldap_admins_groups_name	Administrator group.	P8Administrators
ic_ldap_name	Name of the LDAP directory.	ldap_name

Table 34. Initialization configuration parameters (continued)

Parameters	Description	Default Values
oc_cpe_obj_store_display_name	Display name for the object store to create.	OS10
oc_cpe_obj_store_symb_name	Symbolic name for the object store to create.	OS10
Object store connection oc_cpe_obj_store_conn name	Object store connection name.	objectstore1_connection
site_name	The name of the site.	InitialSite
dc_os_datasource_name	Add the name of the object store database.	FN0S1DS
dc_os_xa_datasource_name	The XA datasource.	FN0S1DSXA
oc_cpe_obj_store_admin_user_groups	Admin user group.	ceadmin
Basic user groups oc_cpe_obj_store_basic_user_groups	An array of users with access to the object store.	
Enable add-ons oc_cpe_obj_store_addons	Specify whether to enable add-ons.	true
List of add-ons oc_cpe_obj_store_addons_list	Add-ons to enable for Content Platform Engine	(Add-on values)
Advanced storage device oc_cpe_obj_store_asa_name	Provide a name for the device	demo_file_system_storage
ASA root directory oc_cpe_obj_store_asa_root_dir_path	The root directory path for the object store storage area.	/opt/ibm/asa/os10_storagearea1
Enable workflow oc_cpe_obj_store_enable_workflow	Specify whether to enable workflow for the object store.	false
Workflow region name oc_cpe_obj_store_workflow_region_name	Specify a name for the workflow region	design_region_name
Workflow region number os_cpe_obj_store_workflow_region_number	Specify the number of the workflow region.	1
Workflow table space os_cpe_obj_store_workflow_data_tbl_space	Specify a table space for the workflow data.	VWDATA_TS

Table 34. Initialization configuration parameters (continued)

Parameters	**Description**	**Default Values**
Workflow index table space oc_cpe_obj_store_workflow_index_tbl_space	Optionally specify a table space for the workflow index.	
Workflow blob table space oc_cpe_obj_store_workflow_blob_tbl_space	Optionally specify a table space for the workflow blob.	
Workflow admin group oc_cpe_obj_store_workflow_admin_group	Designate an LDAP group for the workflow admin group.	P8Administrators
Workflow config group oc_cpe_obj_store_workflow_config_group	Designate an LDAP group for the workflow config group.	P8Administrators
Date time mask oc_cpe_obj_store_workflow_date_time_mask	Default format for date and time	mm/dd/y hh:tt am
Locale oc_cpe_obj_store_workflow_locale	Locale for the workflow	en
PE connection point oc_cpe_obj_store_workflow_pe_conn_point_name	Provide a name for the connection point	pe_conn_os1
Site name css_site_name	The site name for CSS.	Initial Site
Text Search server css_text_search_server_name	Name for the text search server	{{ meta.name }}-css-1
Affinity group affinity_group_name	The name for an affinity group	aff_group
Server status css_text_search_server_status	Status of the CSS text search server.	0
Server mode css_text_search_server_mode	Mode of the CSS text search server.	0
Enable SSL css_text_search_server_ssl_enable	Specify whether to enable SSL for the CSS text search server.	true

Table 34. Initialization configuration parameters (continued)

Parameters	**Description**	**Default Values**
Credentials css_text_sesarch_server_credential	Credentials for the CSS text search server	RNUNEWc=
Host css_text_search_server_host	Host name for the CSS text search server.	{{ meta.name }}-css-svc-1
Port css_text_search_server_port	Port for the CSS text search server.	8199
Object store object_store_name	Name of the associate object store	OS10
Index area index_area_name	Provide a name for the index area.	os1_index_area
Affinity group affinity_group_name	Provide the value that you specified for the associated server affinity group name.	aff_group
Root directory root_dir	The root directory for the index area.	/opt/ibm/indexareas
Maximum index count max_indexes	Specify the maximum number of indexes	20
Maximum object per index max_objects_per_index	Specify the maximum number of objects per index.	10000
Object store object_store_name	Name of the object store for CBR.	OS10
Class class_name	Class name.	Document
Language indexing_languages	Language for indexing	en
Repository ID add_repo_id	For IBM Content Navigator, specify the repository.	demo_repo1
URL add_repo_ce_wsi_url	For IBM Content Navigator, specify the URL for the repository.	http://<hostname>:9080/wsi/FNCEWS40MTOM/
Symbolic name add_repo_os_sym_name	For IBM Content Navigator, specify the symbolic name for the repository	OS10

Table 34. Initialization configuration parameters (continued)

Parameters	Description	Default Values
Display name add_repo_os_dis_name	For IBM Content Navigator,specify the display name for the repository	OS10
Enable workflow add_repo_workflow_enable	Enable workflow for the repository	false
Workflow connection point add_repo_work_conn_pnt	Add the name of the pe connection point that you created and the region that you specified (default is 1)	pe_conn_os1:1
Repository protocol add_repo_protocol	Specify the repository protocol.	FileNetP8WSI
Repository ID add_repo_id	Specify the repository ID.	test_repo2
Repository WSI URL add_repo_ce_wsi_url	Specify the URL.	http://<hostname>:9080/wsi/FNCEWS40MTOM
Repository symbolic name add_repo_os_sym_name	Specify the symbolic name of the repository.	OS3
Display name add_repo_os_dis_name	Specify the display name of the repository.	SOS3
add_repo_workflow_enable	Specify whether to enable workflow.	true
add_repo_work_conn_pnt: "pe_conn_os3:1"	Add the name of the pe connection point that you created and the region that you specified (default is 1).	pe_conn_os3:1
Protocol add_repo_protocol	Specify the protocol for the repository	FileNetP8WSI
Desktop ID add_desktop_id	Add the ID of the IBM Content Navigator desktop	demo
Desktop name add_desktop_name	Add the name of the desktop.	icn_desktop
Description add_desktop_description	Provide a description of the desktop.	This is ICN desktop
Repository ID add_desktop_repo_id	Add the repository ID for the desktop.	demo_repo1

Table 34. Initialization configuration parameters (continued)

Parameters	Description	Default Values
Enable workflow add_desktop_repo_workflow_enable	Enable workflow for the desktop.	false

Verify parameters

Update the custom YAML file to provide the details that are relevant for verifying your FileNet Content Manager and IBM Content Navigator components.

Table 35. Verify configuration parameters

Parameters	Description	Default Values
Object store folder_cpe_obj_store_name	Name of the object store that was created during initialization.	OS10
Folder path folder_cpe_folder_path	The path to the test folder.	/TESTFOLDER
Object store doc_cpe_obj_store_name	Name of the object store that was created during initialization.	OS10
Folder path doc_cpe_folder_name	The path to the test folder for the document test.	/TESTFOLDER
Document title doc_cpe_doc_title	A title for the test document.	test_title
Class name DOC_CPE_class_name	The Content Platform Engine class to use when creating the test document.	Document
Document content doc_cpe_doc_content	The contents of the test document.	This is a simple document test
Document content name doc_cpe_doc_content_name	Name for the document content.	doc_content_name
Object store cbr_cpe_obj_store_name	Name of the object store for CBR test.	OS10
Class cbr_cpe_class_name	Class for the CBR test.	Document
Search string cbr_cpe_search_string	String value should correspond to value for doc_cpe_doc_content.	is a simple
Workflow workflow_cpe_enabled	Specify whether workflow is enabled on the object store.	false

Table 35. Verify configuration parameters (continued)

Parameters	Description	Default Values
PE connection point workflow_cpe_connection_point	If workflow is enabled, provide the PE connection point that you specified for the object store.	pe_conn_os1
IBM Content Navigator repository vc_icn_repository	Specify the name of the IBM Content Navigator repository that you configured.	demo_repo1
Desktop vc_icn_desktop_id	Specify the ID for the IBM Content Navigator desktop that you configured.	demo

(V5.5.3 and earlier) Configuration reference

When you complete the configuration tool screens or edit the appropriate YAML file for deployment, you supply values about your supporting environment and your IBM Cloud Private configuration. Collect the configuration parameters as you set up your environment for the content services container deployment.

Deployment parameters

Provide the details of your IBM Cloud Private environment for the overall deployment parameters.

Table 36. Overall deployment parameters

Description	Configuration Tool	YAML	Your Value
Choose between a new deployment or an upgrade from on-premises to containers.	Deployment type	SC_ECC_DEPLOYMENT_TYPE	
Set a name for the deployment.	Deployment name	N/A	
Provide a description for the deployment.	Deployment description	N/A	
If you want to upgrade Content Platform, it is required that you provide the existing `Content Platform Engine-**.ear` file. By default it is under the ear folder of your FileNet Configuration Manager profile.	Upload Content Platform EAR file	N/A	
Choose IBM Cloud Private.	Cluster type	N/A	
Specify the version of IBM Cloud Private that you are using.	Version	ICP_VERSION	
Provide the URL of the IBM Cloud Private console.	Console URL	ICP_CONSOLE_URL	

Table 36. Overall deployment parameters (continued)

Description	**Configuration Tool**	**YAML**	**Your Value**
The IBM Cloud Private repository name with port to use to pull the Content Platform image.	Repository name with port	KC_REPO	
IBM Cloud Private administrator.	Administrator name	ICP_ADMIN_USER	
The password for the IBM Cloud Private administrator.	Administrator password	ICP_ADMIN_PWD	
Master/API Server hostname or IP address with port.	Master node/API server	SC_K8S_ENVIRONMENT	
IBM Cloud Private infrastructure cluster name.	Cluster name	ICP_CLUSTER_NAME	
The IBM Cloud Private account name.	Account name	ICP_ACCOUNT	
The IBM Cloud Private token needed to gain access to the platform.	Token	SC_K8S_TOKEN	
The IBM Cloud Private cluster context information.	Cluster context	SC_K8S_CLUSTER_CONTEXT	
Namespaces are a way to divide cluster resources between multiple users.	Namespace	SC_ECC_TARGET_NAMESPACE	
Existing persistent volume claim name for IBM Business Automation Configuration Container configuration information.	PVC name for IBM Business Automation Configuration Container	SC_ECC_PVC_NAME	
The IBM Cloud Private job container image name in your same namespace, for example, value:*cluster_CA_domain*:8500/<namespace>/imagename:tagname	Job container image name	K8S_CONTAINER_IMAGE_NAME	

Content Platform Engine parameters

Provide the details that are relevant to your Content Platform Engine and your decisions for the deployment of the container.

Table 37. Content Platform Engine configuration parameters

Description	Configuration Tool	YAML	Your Value
The name that IBM Cloud Private uses to track the Content Platform installation.	Helm release name	RELEASE_NAME	
Content Platform Engine image name to be used, for example: value:<cluster_CA_domain>:8500/ imagename:tagname	Image name	K8S_CONTAINER_IMAGE_NAME	
The persistent volume claim for Content Platform Engine configuration.	Configuration overrides PVC name	CPE_CFG_VOLUME	
The persistent volume claim for Content Platform Engine logs.	Logs PVC name	CPE_LOG_VOLUME	
The persistent volume claim for the Content Platform Engine files.	File PVC name	CPE_FILE_VOLUME	
The persistent volume claim for the IBM Case Manager rules.	ICM rules PVC name	CPE_ICMRULES_VOLUME	
The persistent volume claim for text extraction	Text extraction PVC name	CPE_TEXTEXT_VOLUME	
The persistent volume claim for upgrade/ startup.	Bootstrap PVC name for upgrade	CPE_BOOTSTRAP_VOLUME	
The persistent volume claim for FileNet logs.	FileNet logs PVC name	CPE_FNLOG_VOLUME	
How many Content Platform Engine replicas to deploy.	Replicas	REPLICAS	
The minimum JVM heap setting	Minimum JVM heap	JVM_HEAP_XMS	
The maximum JVM heap setting.	Maximum JVM heap	JVM_HEAP_XMX	
The time zone for the container deployment.	Time Zone for container	TZ	
Choose the licensing model. Required.	License Model	LICENSEMODEL	

Table 37. Content Platform Engine configuration parameters (continued)

Description	Configuration Tool	YAML	Your Value
Specifies a CPU request for the container.	Resource -> CPU request	CPU_REQUEST	
Specify a CPU limit for the container.	Resource -> CPU limit	CPU_LIMIT	
Specify a memory request for the container.	Resource -> Memory request	MEMORY_REQUEST	
Specify a memory limit for the container.	Resource -> Memory limit	MEMORY_LIMIT	
Specify whether to enable auto scaling.	Auto Scale → Enable	AUTO_SCALING_ENABLED	
The upper limit for the number of pods that can be set by the autoscaler. Required.	Auto Scale -> Maximum scale	AUTO_SCALING_MAX	
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.	Auto Scale -> Minimum scale	AUTO_SCALING_MIN	
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 will be used.	Auto Scale -> CPU percentage	AUTO_SCALING_CPU	
Specify whether to use the built-in monitoring capability.	Enable monitoring	KC_MONITORING_ENABLED	
Specify whether to use the built-in logging capability.	Enable logging	KC_LOGGING_ENABLED	

Tip: Any default values that are provided for resource settings are suggestions only. Your resource requirements might differ significantly.

On IBM Cloud Private, you can use the console to obtain the appropriate resource settings for your deployment.

IBM Content Navigator parameters

Provide the details that are relevant to your IBM Content Navigator and your decisions for the deployment of the container.

Table 38. IBM Content Navigator configuration parameters

Description	Configuration Tool	YAML	Your Value
The name which IBM Cloud Private uses to track the IBM Content Navigator installation.	Helm release name	RELEASE_NAME	
IBM Content Navigator image name to be used, example value: <cluster_CA_domain>:8500/imagename:tagname	Image name	IMAGE_NAME	
IBM Content Navigator configuration overrides persistent volume claim name.	Configuration overrides PVC name	ICN_CFG_VOLUME	
IBM Content Navigator logs persistence volume name.	Logs PVC name	ICN_LOGS_VOLUME	
IBM Content Navigator plugins persistence volume name.	Plugins PVC name	ICN_PLUGIN_VOLUME	
IBM Content Navigator viewer cache persistence volume name.	Viewer cache PVC name	ICN_VW_CACHE_VOLUME	
IBM Content Navigator viewer logs persistence volume name.	Viewer logs PVC name	ICN_VS_LOGS_VOLUME	
How many IBM Content Navigator replicas to be deployed in Kubernetes.	Replicas	REPLICAS	
Initial Java heap size in megabytes.	Minimum JVM heap (MB)	JVM_HEAP_XMS	
Maximum Java heap size in megabytes.	Maximum JVM heap (MB)	JVM_HEAP_XMX	
Time Zone For information about the value for this property, see the following web page: https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.	Time Zone for container	TZ	

Table 38. IBM Content Navigator configuration parameters (continued)

Description	Configuration Tool	YAML	Your Value
IBM Content Navigator database schema name.	Schema name	ICNSCHEMA	
IBM Content Navigator table space name.	Table space	ICNTS	
To specify a CPU request for a container.	Resource -> CPU request	CPU_REQUEST	
To specify a CPU limit for a container.	Resource -> CPU limit	CPU_LIMIT	
To specify a Memory request for a container.	Resource -> Memory request	MEMORY_REQUEST	
To specify a memory limit for a container.	Resource -> Memory limit	MEMORY_LIMIT	
The upper limit for the number of pods that can be set by the autoscaler. Required.	Auto Scale -> Maximum scale	AUTO_SCALING_MAX	
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.	Auto Scale -> Minimum scale	AUTO_SCALING_MIN	
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 will be used.	Auto Scale -> CPU percentage	AUTO_SCALING_CPU	
Add monitoring service .	Enable monitoring	KC_MONITORING_ENABLED	
Add logging service.	Enable logging	KC_LOGGING_ENABLED	

Content Search Services parameters

Provide the details that are relevant to your IBM Content Search Services and your decisions for the deployment of the container.

Table 39. IBM Content Search Services configuration parameters

Description	Configuration Tool	YAML	Your Value
The name which IBM Cloud Private uses to track the IBM Content Search Services.	Helm release name	RELEASE_NAME	

Table 39. IBM Content Search Services configuration parameters (continued)

Description	**Configuration Tool**	**YAML**	**Your Value**
IBM Content Search Services image name to be used, example value:<cluster_CA_domain>:8500/imagename:tagname	Image name	IMAGE_NAME	
IBM Content Search Services data persistent volume claim name.	Data PVC name	CSS_DATA_VOLUME	
IBM Content Search Services logs persistence volume name.	Logs PVC name	CSS_LOG_VOLUME	
IBM Content Search Services temp data persistence volume name.	Temp data PVC name	CSS_TEMP_VOLUME	
IBM Content Search Services indexes persistence volume name.	Indexes PVC name	css-icp-indexstore	
Maximum Java heap size in megabytes.	Maximum JVM heap(MB)	JVM_HEAL_XMX	
To specify a CPU request for a container.	Resource -> CPU request	CPU_REQUEST	
To specify a CPU limit for a container.	Resource -> CPU limit	CPU_LIMIT	
To specify a Memory request for a container.	Resource -> Memory request	MEMORY_REQUEST	
To specify a memory limit for a container	Resource -> Memory limit	MEMORY_LIMIT	
Add monitoring service.	Enable monitoring	KC_MONITORING_ENABLED	
Add logging service.	Enable logging	KC_MONITORING_ENABLED	

Content Management Interoperability Services parameters

Provide the details that are relevant to your IBM Content Management Interoperability Services and your decisions for the deployment of the container.

Table 40. IBM Content Management Interoperability Services configuration parameters

Description	Configuration Tool	YAML	Your Value
The name which IBM Cloud Private uses to track the IBM Content Management Interoperability Services	Helm release name	RELEASE_NAME	
IBM Content Management Interoperability Services image name to be used, example value:<cluster_CA_domain>:8500/ imagename:tagname	Image name	IMAGE_NAME	
IBM Content Management Interoperability Services configuration overrides persistent volume claim name	Configuration overrides PVC name	CMIS_CFG_VOLUME	
IBM Content Management Interoperability Services logs persistence volume name.	Logs PVC name	CMIS_LOGS_VOLUME	
How many IBM Content Management Interoperability Services replicas to be deployed in Kubernetes.	Replicas	Replicas	
Initial Java heap size in megabytes.	Minimum JVM heap (MB)	JVM_HEAP_XMS	
Maximum Java heap size in megabytes.	Maximum JVM heap (MB)	JVM_HEAP_XMX	
To specify a CPU request for a container.	Resource -> CPU request	CPU_REQUEST	
To specify a CPU limit for a container	Resource -> CPU limit	CPU_LIMIT	
To specify a memory request for a container.	Resource -> Memory request	MEMORY_REQUEST	
To specify a memory limit for a container.	Resource -> Memory limit	MEMORY_LIMIT	

Table 40. IBM Content Management Interoperability Services configuration parameters (continued)			
Description	**Configuration Tool**	**YAML**	**Your Value**
The upper limit for the number of pods that can be set by the autoscaler. Required..	Auto Scale -> Maximum scale	AUTO_SCALING_MAX	
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.	Auto Scale -> Minimum scale	AUTO_SCALING_MIN	
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 will be used.	Auto Scale -> CPU percentage	AUTO_SCALING_CPU	
Add monitoring service.	Enable monitoring	KC_MONITORING_ENABLED	
Add logging service.	Enable logging	KC_MONITORING_ENABLED	

Logging parameters

Provide the details that are relevant to the logging integration that you selected.

Table 41. Logging configuration parameters			
Description	**Configuration Tool**	**YAML**	**Your Value**
Use one of the following settings to determine for the log shipper: 1: Filebeat 2: Logstash mtlumberjack output for Bluemix logging service.	Log shipper option	MON_LOG_SHIPPER_OPTION	
Logging service endpoint using format host1:port, host2:port, ..., hostN:port. The port for all hosts must be same.	Logging service end point	MON_LOG_SERVICE_ENDPOINT	
Bluemix logging service token.	Logging Bluemix token	MON_BMX_LOGS_LOGGING_TOKEN	

Table 41. Logging configuration parameters (continued)

Description	Configuration Tool	YAML	Your Value
GUID that identifies the space or organization domain where the metrics are stored. For more information about this property, see the following web page:https://pages.github.ibm.com/metrics-service/collectd/collectd-monitoring_plugin.html	Logging Bluemix space ID	MON_BMX_SPACE_ID	

Monitoring parameters

Provide the details that are relevant to the monitoring integration that you selected.

Table 42. Monitoring configuration parameters

Description	Configuration Tool	YAML	Your Value
Provide the monitoring metrics option. Specify 2 for IBM Cloud monitoring service, 0 for extra monitoring service.	Monitor metrics writer option	MON_METRICS_WRITER_OPTION	
Provide the monitoring metrics service endpoint using the format host:port.	Monitor metrics service end point	MON_METRICS_SERVICE_ENDPOINT	
Bluemix metrics group name ended with ".", which is used for metrics prefix.	Monitor metrics Bluemix group	MON_BMX_GROUP	
GUID that identifies the space or organization domain where the metrics are stored. For more information about this property, see the following web page: https://pages.github.ibm.com/metrics-service/collectd/collectd-monitoring_plugin.html.	Monitor metrics Bluemix scope ID	MON_BMX_METRICS_SCOPE_ID	
Bluemix API key, which is used for all the RESTful APIs.	Monitor metrics Bluemix API key	MON_BMX_API_KEY	

Initialization parameters

If you choose the initialization option in the deployment, you must provide the details that are relevant to the initialization of your container environment.

Table 43. Initialization configuration parameters

Description	Configuration Tool	YAML	Your Value
Admin user for deployments. This will be the admin user for both Content Platform and Content Navigator if selected.	Admin user	SC_ECC_USERNAME	
Admin password for deployments. This will be the admin user for both Content Platform and Content Navigator if selected.	Admin password	SC_ECC_PASSWORD	
Enter the domain name.	Domain name	DOMAIN_NAME	
Enter the encryption key.	Encryption key	ENCRYPTION_KEY	
Enter the name of the domain admin group.	Domain Admin Group	IC_LDAP_ADMINS_GROUPS_NAME	
The display name for the Content Platform Engine object store.	Content Platform object store display name	OC_CPE_OBJ_STORE_DISPLAY_NAME	
Specify the connection name for the object store.	Connection name	OC_CPE_OBJ_STORE_CONN	
Specify the non-XA data source name for the object store.	Object store data source name	DC_OS_DATASOURCE_NAME	
Specify the XA data source name for the object store.	Object store data source XA name	DC_OS_XA_DATASOURCE_NAME	
	Enable workflow		
Enter the table space name for the object store data.	Object store Data Table Space	OC_CPE_OBJ_STORE_WORKFLOW_DATA_TBL_SPACE	
Enter the name of the Workflow Admin Group.	Workflow Admin Group	OC_CPE_OBJ_STORE_WORKFLOW_ADMIN_GROUP	
Enter the name of the Workflow Config Group.	Workflow Config Group	OC_CPE_OBJ_STORE_WORKFLOW_CONFIG_GROUP	
Indicate whether to enable Addons.	Enable AddOns	OC_CPE_OBJ_STORE_ADDONS	

Table 43. Initialization configuration parameters (continued)

Description	Configuration Tool	YAML	Your Value
(For IBM Content Navigator) Specify a name for the desktop.	Desktop name	ADD_DESKTOP_NAME	
(For IBM Content Navigator) Specify a description for the desktop.	Desktop description	ADD_DESKTOP_DESCRIPTION	
(For IBM Content Navigator) Add a name for the repository.	Repository display name	ADD_REPO_NAME	
(For IBM Content Navigator) Choose the object store.	Object store	ADD_REPO_OS_SYM_NAME	

LDAP parameters

Provide the details that are relevant for the LDAP configuration of your container environment.

Table 44. LDAP configuration parameters

Description	Configuration Tool	YAML	Your Value
Specify your directory service (LDAP) provider.	Directory service provider	LC_SELECTED_LDAP_TYPE	
Enter the short name,long name or the IP address of the directory server host in a format that can be resolved from your web application server.	Directory service provider host name	LC_LDAP_SERVER	
Enter the port number that is configured on the directory server host for communicating with the directory server. The default port is 389. If you use SSL to communicate with the directory server the default port is 635.	Directory service server port number	LC_LDAP_PORT	
Enter the fully distinguish name of the LDAP bind user. The application server uses this user account to bind to TDS to authenticate user credentials. This account name must be unique user across all realms.	Directory service server bind user name	LC_LDAP_BIND_DN	

Table 44. LDAP configuration parameters (continued)

Description	Configuration Tool	YAML	Your Value
Enter Directory service server bind user password.	Directory service server bind user password	LC_LDAP_BIND_PASSWORD	
A fully qualified distinguish name that is the starting point to search the directory for users who can log in to the repository. The user base distinguish name and the user filter together define the search criteria that determine the set of eligible users. To limit the set size, specify a directory subtree as the user base distinguish name, for example ou=FileNetUsers,DC=FNCE,DC=Region01,DC=FileNet,DC=com.	Base entry distinguished name (repository)	LC_LDAP_BASE_DN	
The filter by which the bind user searches for groups in Tivoli Directory Server. The group filter & the base distinguish name together define the search criteria that determine the set of eligible groups.	Directory service server group filter	LC_GROUP_FILTER	
The attribute that determines the type of information that a user enters to log on to Active Directory. The attribute can be any property on the LDAP user account, such as serial ID, email address, or user name.	Directory service server user ID map	LC_LDAP_USER_NAME_ATTRIBUTE	
An attribute in the directory server entry that identifies the group.	Directory service server group ID map	LC_LDAP_GROUP_NAME_ATTRIBUTE	

Database parameters

Provide the details that are relevant for the database configuration of your container environment.

Table 45. Db2 database configuration parameters

Description	Configuration Tool	YAML	Your Value
Select the JDBC driver for your database type.	JDBC driver name	DC_DATABASE_TYPE	
Upload the JDBC driver from the database server.	Upload JDBC driver		
The JNDI name of the non-XA JDBC data source associated with the global configuration table space or database. The name must be unique.	(For GCD) JDBC data source name	DC_COMMON_GCD_DATASOURCE_NAME	
The JNDI name of the XA JDBC data source associated with the global configuration table space or database. The name must be unique.	(For GCD) JDBC XA data source name	DC_COMMON_GCD_XA_DATASOURCE_NAME	
The host name of the server where the database software is installed.	(For GCD) Database server name	DC_DB2_HADR_DATABASE_SERVERNAME	
Enter the database port number.	Database port number	DC_DB2_HADR_DATABASE_PORT	
Enter the database name.	Database name	DC_DB2_HADR_OS_DATABASE_NAME	
Enter the database user name.	Database user name	DC_COMMON_DATABASE_INSTANCE_USER	
Enter the database password.	Database password	DC_COMMON_DATABASE_INSTANCE_PASSWORD	
Enter the standby server name.	Standby server name	DC_HADR_STANDBY_SERVERNAME	
Enter the standby port number.	Standby port	DC_HADR_STANDBY_PORT	
Specify the validation timeout entry.	Validation timeout	DC_HADR_VALIDATION_TIMEOUT	

Table 45. Db2 database configuration parameters (continued)

Description	**Configuration Tool**	**YAML**	**Your Value**
Specify the time in seconds between connection attempts made by the automatic client reroute if the primary connection to the server fails.	Retry interval for client reroute	DC_HADR_RETRY_INTERVAL_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.	Maximum retries for client reroute	DC_HADR_MAX_RETRIES_FOR_CLIENT_REROUTE	
(For object store 1) The JNDI name of the non-XA JDBC data source associated with the global configuration table space or database. The name must be unique.	JDBC data source name	DC_COMMON_GCD_DATASOURCE_NAME	
(For object store 1) The JNDI name of the XA JDBC data source associated with the global configuration table space or database. The name must be unique.	JDBC XA data source name	DC_COMMON_GCD_XA_DATASOURCE_NAME	
(For object store 1) The host name of the server where the database software is installed.	Database server name	DC_DB2_HADR_DATABASE_SERVERNAME	
(For object store 1) Enter the database port number.	Database port number	DC_DB2_HADR_DATABASE_PORT	
(For object store 1) Enter the database user name.	Database user name	DC_COMMON_DATABASE_INSTANCE_USER	
(For object store 1) Enter the database password.	Database password	DC_COMMON_DATABASE_INSTANCE_PASSWORD	

Table 45. Db2 database configuration parameters (continued)

Description	**Configuration Tool**	**YAML**	**Your Value**
(For object store 1) (pureScale or HADR) Provide the name of the standby server.	Standby server name	DC_HADR_STANDBY_SERVERNAME	
(For object store 1) (pureScale or HADR) Enter the standby server port.	Standby port	DC_HADR_STANDBY_PORT	
(For object store 1) (pureScale or HADR) Specify the validation timeout entry.	Validation timeout	DC_HADR_VALIDATION_TIMEOUT	
(For object store 1) (pureScale or HADR) Specify the time in seconds between connection attempts made by the automatic client reroute if the primary connection to the server fails.	Retry interval for client reroute	DC_HADR_RETRY_INTERVAL_FOR_CLIENT_REROUTE	
(pureScale or HADR) 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.	Maximum retries for client reroute	DC_HADR_MAX_RETRIES_FOR_CLIENT_REROUTE	
(For IBM Content Navigator) Enter the JNDI name of the non-XA JDBC data source that you want to create to communicate with the IBM Content Navigator database.	IBM Content Navigator data source name	DC_COMMON_ICN_DATASOURCE_NAME	
(For IBM Content Navigator) The host name of the server where the database software is installed.	Database server name	DC_DB2_HADR_DATABASE_SERVERNAME	
(For IBM Content Navigator) Enter the database port number.	Database port number	DC_DB2_HADR_DATABASE_PORT	

Table 45. Db2 database configuration parameters (continued)

Description	Configuration Tool	YAML	Your Value
(For IBM Content Navigator) Enter the name of the database you created to store the IBM Content Navigator configuration data.	Database name	DC_DB2_HADR_ICN_DATABASE_NAME	
(For IBM Content Navigator) Enter the name of the database user that this data source will use to connect to the database. The user must have INSERT, UPDATE, DELETE and SELECT authority on the IBM Content Navigator configuration tables.	Database user name	DC_COMMON_DATABASE_INSTANCE_USER	
(For IBM Content Navigator) Enter the database password.	Database password	DC_COMMON_DATABASE_INSTANCE_PASSWORD	
(For IBM Content Navigator) (pureScale or HADR) Enter the standby server name.	Standby server name	DC_HADR_STANDBY_SERVERNAME	
(For IBM Content Navigator) (pureScale or HADR) Enter the port number for the standby server	Standby port	DC_HADR_STANDBY_PORT	
(For IBM Content Navigator) (pureScale or HADR) Specify the validation timeout entry.	Validation timeout	DC_HADR_VALIDATION_TIMEOUT	
(For IBM Content Navigator) (pureScale or HADR) Specify the time in seconds between connection attempts made by the automatic client reroute if the primary connection to the server fails.	Retry interval for client reroute	DC_HADR_RETRY_INTERVAL_FOR_CLIENT_REROUTE	

Table 45. Db2 database configuration parameters (continued)

Description	Configuration Tool	YAML	Your Value
(For IBM Content Navigator) (pureScale or HADR) 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.	Maximum retries for client reroute	DC_HADR_MAX_RETRIES_FOR_CLIENT_REROUTE	

Table 46. Oracle database configuration parameters

Description	Configuration Tool	YAML	Your Value
Select the JDBC driver for your database type.	JDBC driver name	DC_DATABASE_TYPE	
Upload the driver from the database server.	Upload JDBC driver		
Enter the Oracle JDBC URL.	Oracle JDBC URL	DC_ORACLE_GCD_JDBC_URL	
The JNDI name of the non-XA JDBC data source associated with the global configuration table space or database. The name must be unique.	(For GCD) JDBC data source name	DC_COMMON_GCD_DATASOURCE_NAME	
The JNDI name of the XA JDBC data source associated with the global configuration table space or database. The name must be unique.	(For GCD) JDBC XA data source name	DC_COMMON_GCD_XA_DATASOURCE_NAME	
The database user name by which Content Platform Engine accesses the global configuration database or an object store table space. You defined this user as cpe_db_user.	Database user name	DC_COMMON_DATABASE_INSTANCE_USER	
Enter the database password.	Database password	DC_COMMON_DATABASE_INSTANCE_PASSWORD	

Table 46. Oracle database configuration parameters (continued)

Description	**Configuration Tool**	**YAML**	**Your Value**
(For object store 1) Enter the Oracle object store JDBC URL.	Oracle JDBC URL	DC_ORACLE_OS_JDBC_URL	
(For object store 1) The JNDI name of the non-XA JDBC data source associated with the object store table space or database. The name must be unique.	JDBC data source name	DC_COMMON_OS_DATASOURCE_NAME	
(For object store 1) The JNDI name of the XA JDBC data source associated with the global configuration table space or database. The name must be unique.	JDBC XA data source name	DC_COMMON_OS_XA_DATASOURCE_NAME	
(For object store 1) The database user name by which Content Platform Engine accesses the global configuration database or an object store table space. You defined this user as cpe_db_user.	Database user name	DC_COMMON_DATABASE_INSTANCE_USER	
(For object store 1) Enter the database password.	Database password	DC_COMMON_DATABASE_INSTANCE_PASSWORD	
(For IBM Content Navigator) Enter the Oracle object store JDBC URL.	Oracle JDBC URL	DC_ORACLE_ICN_JDBC_URL	
(For IBM Content Navigator) Enter the JNDI name of the non-XA JDBC data source that you want to create to communicate with the IBM Content Navigator database.	IBM Content Navigator data source name	DC_COMMON_ICN_DATASOURCE_NAME	

Table 46. Oracle database configuration parameters (continued)

Description	**Configuration Tool**	**YAML**	**Your Value**
(For IBM Content Navigator) Enter the name of the database user that this data source will use to connect to the database. The user must have INSERT, UPDATE, DELETE and SELECT authority on the IBM Content Navigator configuration tables.	Database user name	DC_COMMON_DATABASE_INSTANCE_USER	
(For IBM Content Navigator) Enter the database password.	Database password	DC_COMMON_DATABASE_INSTANCE_PASSWORD	