OpenText™ Documentum™ Content Management

Cloud Deployment Guide

Deploy and configure OpenText Documentum Content Management (CM) on certified cloud platforms.

EDCSYCD250400-IGD-EN-02

OpenText™ Documentum™ Content Management
Cloud Deployment Guide
EDCSYCD250400-IGD-EN-02
Rev.: 2026-Feb-20

This documentation has been created for OpenText™ Documentum™ Content Management CE 25.4.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

Overview

As OpenText Documentum CM advances towards the cloud world, it is also designed to guide you to advance your journey towards the cloud world by providing you with road maps to adopt new cloud deployments, while continuing to support legacy environments and have hybrid implementations. The move to the cloud world is driven by the following key capabilities for you to:

- reduce the high operating costs to develop, manage and maintain on-premises applications
- avoid end user adoption issues caused by slow performance and lengthy deployment time lines
- gain access to extensive resources to support Enterprise Information Management (EIM) applications
- deploy EIM applications and grow as needed to scale to your business needs

This guide is intended for system, repository, and cloud administrators, application programmers, and any other user who wishes to obtain an understanding of deploying OpenText Documentum CM on different cloud platforms. Users who read this guide must have working knowledge of OpenText Documentum CM components and orchestration in clusters such as Kubernetes using container images and Helm, and have an understanding of databases and services of different cloud platforms.

With evidence that the cloud is the future for data, and that it is imminent that enterprise workloads will run in the cloud, OpenText encourages you to choose the cloud over an on-premises solution.

OpenText Documentum CM offers a single All-In-One Helm package for the supported containers. This approach defines a common layout within the OpenText Documentum CM solutions and includes all containers with the ability to deploy and upgrade using a single Helm package.

You can open the `documentum/documentum-components.yaml` file in the single Helm package to enable or disable products and components. For example, OpenText Documentum CM Server, OpenText Documentum CM client, and so on.

After providing the appropriate values according to your business requirement in all the required YAML files, you can choose to deploy multiple containers along with OpenText Documentum CM Server with one Helm install command. Similarly, you can use one Helm upgrade command to upgrade multiple containers.

Chapter 2

Prerequisites for deploying and configuring OpenText Documentum CM on cloud platforms

This documentation provides the information about downloading and configuring Helm package, cluster, container image, and Helm charts. In addition, you can find information to set up and configure Microsoft Azure, Amazon Web Services, and Google Cloud Platform cloud platforms.

2.1 Downloading Helm and Helm charts, container images, and configuring cloud platform-specific requirements

Kubernetes is a portable, extensible open-source platform orchestration engine for automating deployment, scaling, and management of containerized applications. Kubernetes can be considered as a container platform, microservices platform, and portable cloud platform. It provides a container-centric management environment.

OpenShift is an enterprise-ready, subscription-based platform orchestration engine for automating deployment, scaling, and management of containerized applications. OpenShift is a container platform that provides capabilities to manage advanced clusters.

You can use the containerized deployment using the OpenText Documentum CM container images and OpenText Documentum CM Helm charts that are packaged with the release.

The product *Release Notes* document contains detailed information about the list of applications and its supported versions.

To download and configure Helm package, cluster, container images, and Helm charts:

1. Download and configure the supported version of the Helm package from the Helm website.

   The product *Release Notes* document contains information about the supported versions.

   Helm documentation contains detailed information.

2. If you are using Kubernetes, download and configure the Kubernetes application from the Kubernetes website.

   Kubernetes documentation contains detailed information.

3. If you are using Red Hat OpenShift, download and configure the Red Hat OpenShift application from the Red Hat website.

   Red Hat OpenShift documentation contains detailed information.

   Note: Red Hat OpenShift is supported on Microsoft Azure and Amazon Web Services.

4. Set up the supported database servers for all products that need a database.

   The product *Release Notes* document contains information about the supported databases.

   Note: The PostgreSQL and Oracle database clients are packaged with the Documentum CM Server container image.

5. Optional If you are configuring the PostgreSQL container database, provide the PostgreSQL image details in the `db` section of `documentum/dockerimages-values.yaml` file.

6. Optional Download the Fluentd Docker image from OpenText Container Registry.

   For information about the instructions, see step 8.

7. Optional Upload the Fluentd Docker image.

   Make sure that you configure the Fluentd Docker image according to your requirement.

8. Download the following container images from OpenText Container Registry:

   - Oracle Linux container image for all components except for the following:
     – OpenText Documentum CM Foundation SOAP API
     – OpenText Documentum CM Foundation REST API
     – OpenText Documentum CM Foundation CMIS API
     – OpenText Documentum CM Administrator
     – OpenText Content Connect
   - Alpine Linux container image for the following components:
     – OpenText Documentum CM Foundation SOAP API
     – OpenText Documentum CM Foundation REST API
     – OpenText Documentum CM Foundation CMIS API
     – OpenText Documentum CM for Microsoft 365
     – OpenText Content Connect
   - Oracle Linux Slim container image for OpenText Documentum CM Administrator.

Note: OpenText Documentum CM for Microsoft 365 Notification service uses the Oracle Linux container image.

Using container images

Container images in the OpenText Container Registry are identified by the base container image name and a tag (version). You can obtain container images using two types of tags:

- `YY.Q.#`

  In a tag of this style, the dot-separated numbers represent the year, quarter, and point release. This type of image tag identifies a specific point-in-time release with the patches and configurations that were available at the time that the image was built.

  Use a `<YY.Q.#>` tag to specify the exact version number of the image you want to download. For example, if you download a container image of `ot-dctm-server:<YY.Q.#>` using the `docker pull registry.opentext.com/ot-dctm-server:<YY.Q.#>` command, you obtain the specific version of the `Documentum CM Server` container image that you specify, even if more recent images are available.

- `YY.Q`

  In a tag of this style, the dot-separated numbers represent the year and quarter. This type of tag identifies the latest version of a container image available in the OpenText Container Registry.

  Use a `<YY.Q>` tag to obtain the latest image available that has the most recent patches and configurations. For example, to download the latest available version of the `Documentum CM Server` container image, use a `docker pull registry.opentext.com/ot-dctm-server:<YY.Q>` command.

Notes

- OpenText recommends that you use the `<YY.Q>` tag in your `docker pull` command to make sure that you have the latest available image including all patches available at the time the image was built.
- In special cases, OpenText can advise you to use an image with a `<YY.Q.#>` tag for testing or other purposes.

To download the container images from OpenText Container Registry:

a. Run the following command to sign in to OpenText Container Registry:

```
docker login registry.opentext.com
```

   When prompted, provide your OpenText My Support login credentials.

b. Run the following command to download the container image(s):

```
docker pull registry.opentext.com/<image_name>:<image_tag>
```

The following container images are available for download:

Product or component name	Image name	Image tag
OpenText Documentum CM Server	`ot-dctm-server`	25.4.0 or 25.4
OpenText Documentum CM Server	`ot-dctm-fluentd`	25.4.0 or 25.4
OpenText Documentum CM Server	`ot-dctm-dsis`	25.4.0 or 25.4
OpenText Documentum CM client	`ot-dctm-client-installer`	25.4.0 or 25.4
OpenText Documentum CM client	`ot-dctm-client-config`	25.4.0 or 25.4
OpenText Documentum CM client	`ot-dctm-client-classic`	25.4.0 or 25.4
OpenText Documentum CM client	`ot-dctm-client-smartview`	25.4.0 or 25.4
OpenText Documentum CM client	`ot-dctm-client-ijms`	25.4.0 or 25.4
OpenText Documentum CM client	`ot-dctm-client-mobile`	25.4.0 or 25.4
OpenText Documentum CM client	`ot-dctm-client-rest`	25.4.0 or 25.4
Admin Console	`ot-dctm-admin-console`	25.4.0 or 25.4
OpenText Documentum CM Foundation CMIS API	`ot-dctm-cmis`	25.4.0 or 25.4
OpenText Documentum CM Foundation SOAP API	`ot-dctm-dfs`	25.4.0 or 25.4
OpenText Documentum CM Foundation REST API	`ot-dctm-rest`	25.4.0 or 25.4
OpenText Documentum CM Administrator	`ot-dctm-admin`	25.4.0 or 25.4
OpenText Documentum CM Administrator	`ot-dctm-admin-tomcat`	25.4.0 or 25.4
OpenText Documentum CM Records Manager	`ot-dctm-records`	25.4.0 or 25.4
OpenText Documentum CM Records Manager	`ot-dctm-records-darinstallation`	25.4.0 or 25.4
OpenText Documentum CM Records Queue Manager	`ot-dctm-rqm`	25.4.0 or 25.4
OpenText Documentum CM Workflow Designer	`ot-dctm-workflow-designer`	25.4.0 or 25.4

Product or component name	Image name	Image tag
OpenText Documentum CM Advanced Workflow	`ot-dctm-bpm-installer`	25.4.0 or 25.4
OpenText Documentum CM Advanced Workflow	`ot-dctm-xda`	25.4.0 or 25.4
OpenText Documentum CM Advanced Workflow	`ot-dctm-bps`	25.4.0 or 25.4
Documentum xPlore	`ot-dctm-content-fetcher`	25.4
Documentum xPlore	`ot-dctm-search-parser`	25.4
Documentum xPlore	`dctm-xplore-cps`	22.1.14
Documentum xPlore	`dctm-xplore-indexagent`	22.1.14
Documentum xPlore	`dctm-xplore-indexserver`	22.1.14
OpenText Documentum CM Messaging Service	`ot-dctm-dms`	25.4.0 or 25.4
OpenText Documentum CM Reports	`ot-dctm-reports-client`	25.4.0 or 25.4
OpenText Documentum CM Reports	`ot-dctm-reports-base`	25.4.0 or 25.4
OpenText Documentum CM Reports	`ot-dctm-reports-installer`	25.4.0 or 25.4
OpenText Documentum CM for Microsoft 365	`ot-dctm-smartviewm365`	25.4.0 or 25.4
OpenText Documentum CM for Microsoft 365	`ot-dctm-smartviewm365customjar`	25.4.0 or 25.4
OpenText Documentum CM for Microsoft 365	`ot-dctm-smartviewm365-ns`	25.4.0 or 25.4
OpenText Documentum CM Online Editing Service	`ot-dctm-oesconnector`	25.4.0 or 25.4
OpenText Content Connect	`ot-dctm-content-connect`	25.4.0 or 25.4
OpenText Content Connect	`ot-dctm-content-connect-dbinit`	25.4.0 or 25.4
OpenText Documentum Connector for Core Share	`ot-dctm-dcc-dbschema`	25.4 or 25.4.2
OpenText Documentum Connector for Core Share	`ot-dctm-dcc`	25.4 or 25.4.2
OpenText Documentum Connector for Core Share	`ot-dctm-dcc-darinitcontainer`	25.4 or 25.4.2

Product or component name	Image name	Image tag
OpenText Documentum Archive Services for SAP Solutions	`ot-dctm-assap`	25.4.0 or 25.4
OpenText Documentum Archive Services for SAP Solutions ILM	`ot-dctm-assap-ilm`	25.4.0 or 25.4
OpenText Documentum Content Services for SAP Solutions	`ot-dctm-cssap`	25.4.0 or 25.4
Independent Java Method Server	`ot-dctm-ijms`	25.4.0 or 25.4
OpenText Content Aviator	`ot-dctm-dcis`	25.4.0 or 25.4
OpenText Directory Services	`otds-server`	25.4.0
OpenText AppWorks Gateway	`otawg`	25.4.0 or 25.4
OpenText AppWorks Gateway	`otawg-init`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-amqp`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-asset`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-config`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-highlight`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-markup`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-publication`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-init-otds`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-publisher`	25.4.0 or 25.4
OpenText Intelligent Viewing	`otiv-viewer`	25.4.0 or 25.4
Base Tomcat	`ot-dctm-tomcat`	25.4.0 or 25.4

Note: The `ot-dctm-admin-tomcat` image (Oracle Linux Slim) is required only for Documentum Administrator. "Decoupling product image from base Tomcat image" on page 217 contains detailed information.

The `ot-dctm-tomcat` image (Alpine Linux) is required only for the following components:

- OpenText Documentum CM client
- OpenText Documentum CM Foundation CMIS API
- OpenText Documentum CM Foundation SOAP API
- OpenText Documentum CM Foundation REST API
- OpenText Documentum CM Records Manager
- OpenText Documentum CM for Microsoft 365

The following list of images must be downloaded from their respective websites:

Product or component name	Image name	Version
OpenText Content Aviator	`activemq-artemis`	2.41.0

9. Run the following command to upload the container image(s) to your container registry:

```
docker push <image>
```

10. Download the Helm charts from OpenText Container Registry and extract them into a temporary location. Then, do the following:

    a. Run the following command to add the OpenText Helm repository to the Helm's list of repositories:

    ```
    helm repo add opentext https://registry.opentext.com/helm --username
    <user@example.com> --password <password>
    ```

    where <*user@example.com*> is your My Support logon ID, and <*password*> is the password of your My Support logon ID.

    b. Refresh your Helm repository information using the following command:

    ```
    helm repo update
    ```

    c. Run the following command to obtain the version of the required `documentum` Helm chart:

    ```
    helm pull opentext/documentum --version=<##.#.#>
    ```

    where <##.#.#> is the version of the Helm chart.

    For example:

    ```
    helm pull opentext/documentum --version=25.4.0
    ```

    d. After you obtain the Helm chart ZIP file, upload it to the computer that you will use to run the Helm commands.

    e. Run the following command to extract the contents of the ZIP file:

    ```
    tar xvf documentum-<##>.<#>.<#>.tgz
    ```

11. Optional You can create a configuration file for each environment that you deploy. The values in a configuration file overrides the values of `documentum/values.yaml`. You can find a basic template of the `config` file in `documentum/config/configuration.yml`. If you are using this configuration file, make sure that you pass this `configuration.yml` in the Helm deployment commands.

12. Before deploying the OpenText Content Connect pod, you must register the Content Connect applications in Microsoft Entra admin center.

    Log in to the Microsoft Entra admin center.

    Go to App registrations

    a. Click **New Registration**.

    b. Specify the name of the application.

    c. Click **Register**.

    Go to Manage > Authentication

    a. Select **Accounts in this organizational directory only (Single tenant)**.

    b. Click **Add a Platform**.

    c. In configure platforms, click **Web**.

    d. Specify the **Redirect URIs**.

       For example:

       ```
       https://<cc server host>:<port>/cc/ui/templates/msgraphdialog.html
       ```

    e. In **Implicit grant and hybrid flows**, select the following check boxes:

       - **Access token (used for implicit flows)**
       - **ID token (used for implicit and hybrid flows)**

    f. Click **Configure**.

    g. Go to **Overview** and copy `Application (client) ID` and `Directory (tenant) ID`.

       Note: You must provide the values for `client ID` and `tenant ID` in the `documentum/config/configuration.yml` file in `contentconnect` section while deploying the Content Connect pod.

App registration page components	Helm chart variable
Application (client) ID	`clientId`
Directory (tenant) ID	`tenantid`

    Go to Manage > Certificates & secrets

    a. Click **New client secret**.

b. Specify the description in **Description**, and click **Add**.

c. Copy the added value of the secret and save.

> **Note:** Registered app client secret value from Microsoft Entra admin center must be provided for the `clientSecret` variable in the `documentum/config/passwords.yaml` or `documentum/config/passwords_vault.yaml` or `documentum/config/passwords_k8api.yaml` file in `contentconnect` section while deploying the Content Connect pod.

Go to Manage > API permissions

a. Click **Add a permission**.

b. Select **Microsoft Graph**.

c. Select **Delegated permissions**, and select the following delegated permissions:

```
Mail.read
offline_access
Mail.Read.Shared
Mail.ReadWrite
Mail.ReadWrite.Shared
```

d. Click **Add permissions**.

e. Administrator must select **Grant admin consent** to grant a permission.

Go to Manage > Expose an API

a. Click **Add** next to the **Application ID URI**.

b. On the **Edit application ID URI** page, in **Application ID URl**, specify the scope name using the following format:

```
api://<cc server host>:<port>/<auto generated id>
```

For example:

```
api://banddqa902.otxlab:8443/988beae6-8c4b-46ab-9013-22f8e9033355
```

c. Click **Save**.

d. In the **Add a scope** > **Add a client application** section, for **Client ID**, add the following globally unique identifier (GUID):

- Microsoft Office:

  `d3590ed6-52b3-4102-aeff-aad2292ab01c`

- Office on the web:

  `93d53678-613d-4013-afc1-62e9e444a0a5`

- Outlook on the web:

  `bc59ab01-8403-45c6-8796-ac3ef710b3e3`

13. Before deploying the OpenText Documentum CM for Microsoft 365 pod, you must register the OpenText Documentum CM for Microsoft 365 applications in Microsoft Entra admin center.

    Log in to the Microsoft Entra admin center.

    Go to App registrations

    a. Click **New Registration**.

    b. Specify the name of the application.

    c. Click **Register**.

    Go to Manage > Authentication

    a. Select **Accounts in this organizational directory only (Single tenant)**.

    b. Go to **Overview** and copy the `Application (client) ID` and `Directory (tenant) ID`.

       Note: You must provide the values for the following fields in the `documentum/values.yaml` file variables while deploying the OpenText Documentum CM for Microsoft 365 pod.

App registration page components	Helm chart variable
Application (client) ID	`msClientId`
Directory (tenant) ID	`mstenantid`

    c. Go to **Authentication** > **Platform configurations** > **Add a platform** and select a **Single Page Application**.

    d. Select **Redirect URIs** and provide the URL:

       `https://<ingress>:<port>/SmartViewM365/ui`

    e. In **Implicit grant and hybrid flows**, select the following check boxes:

       - **Access token (used for implicit flows)**
       - **ID token (used for implicit and hybrid flows)**

    f. Click **Configure**.

    Go to Manage > Certificates & secrets

    a. Click **New client secret**.

    b. Specify the description in **Description**, and click **Add**.

    c. Copy the added value of the secret and save.

Note: Registered app client secret value from Microsoft Entra admin center must be provided for the `msclientsecret` variable in the `documentum/config/passwords.yaml` or `documentum/config/passwords_vault.yaml` or `documentum/config/passwords_k8api.yaml` file in `smartviewm365` section while deploying the OpenText Documentum CM for Microsoft 365 pod.

Go to Manage > API permissions

a. Click **Add a permission**.

b. Select **Microsoft Graph**.

c. Select **Delegated permissions**, and select the following delegated permissions:

```
Files.Read
Files.Read.All
Files.ReadWrite
Files.ReadWrite.All
Mail.read
offline_access
User.Read
TeamMember.Read.All
```

d. Select **Application permissions**, and select the following application permissions:

```
Directory.Read.All
Directory.ReadWrite.All
GroupMember.Read.All
User.Read.All
User.ReadBasic.All
TeamMember.Read.All
```

e. Select **Microsoft SharePoint**.

f. Select **Delegated permissions**, and select the following delegated permissions:

```
AllSites.Read
AllSites.Write
MyFiles.Read
MyFiles.Write
User.Read.All
User.ReadWrite.All
```

g. Click **Add permissions**.

h. Select **Grant admin consent** to grant all the permissions.

Go to Manage > Expose an API

a. Click **Add** next to the **Application ID URI**.

b. On the **Edit application ID URI** page, in **Application ID URI**, specify the scope name using the following format:

```
api://<ingress>:<port>/<auto generated id>
```

For example:

```
api://banddqa902.otxlab:8443/988beae6-8c4b-46ab-9013-22f8e9033355
```

c. Click **Save**.

d. Click **Add a scope**.

e. In the **Add a scope** screen, specify the **Scope name**, **Admin consent display name**, **Admin consent description** and click **Add scope**.

f. Click **Add a client application** section, for **Client ID**, add the following globally unique identifier (GUID):

```
1fec8e78-bce4-4aaf-ab1b-5451cc387264
5e3ce6c0-2b1f-4285-8d4b-75ee78787346
```

14. You must have Microsoft 365 subscription before deploying the OpenText Documentum CM for Microsoft 365 pod.

15. Before deploying the Documentum Connector for Core Share application, you must register the mail service on Microsoft Entra admin center.

    a. Sign in to the Microsoft Entra admin center and click **App registrations**.

    b. On the **App registrations** page, perform the following:

       i. Click **New registration**.

       ii. Specify the name of the application.

       iii. Click **Register**.

    c. On the **App registrations** page, click the registered application.

    d. On the registered application page, perform the following:

       Go to Manage > Authentication

       i. Select **Accounts in this organizational directory only (Single tenant)**.

       ii. Click **Configure**.

       iii. Go to **Overview** and copy `Application (client) ID` and `Directory (tenant) ID`.

       Note: You must provide the value of `mailservice.properties` for the `clientId` and `tenantid` in the `documentum/config/configuration.yml` file and encrypted `client secret` in the `documentum/config/passwords.yaml` or `documentum/config/passwords_vault.yaml` or `documentum/config/passwords_k8api.yaml` file while deploying the Documentum Connector for Core Share pod.

App registration page components	Helm chart variable
Application (client) ID	`clientId`
Directory (tenant) ID	`tenantid`

Go to Manage > Certificates & secrets

i. Click **New client secret**.

ii. Specify the description in **Description**, and click **Add**.

iii. Copy the added value of the secret and save.

> **Note:** You must provide the value of `mailservice.properties` for the `clientID` in the `documentum/config/configuration.yml` file and `client secret` variable in the `documentum/config/passwords.yaml` or `documentum/config/passwords_vault.yaml` or `documentum/config/passwords_k8api.yaml` file while deploying the Documentum Connector for Core Share pod.

Go to Manage > API permissions

i. Click **Add a permission**.

ii. Select **Microsoft Graph**.

iii. Select **Application permissions** and select the following permissions:

```
Mail.ReadWrite
Mail.send
```

iv. Select **Delegated permissions**, and select the following delegated permissions:

```
User.Read
```

v. Select **Grant admin consent** to grant a permission.

> **Note:** You must have Microsoft 365 subscription with permission to add or modify the Microsoft Entra admin center, and grant application or user permissions listed in the **App registrations**.

16. To use the OpenText Documentum CM Messaging Service pod, you must install Branch Office Caching Services in a virtual machine (VM).

    OpenText recommends that you configure Branch Office Caching Services in the pull mode.

    For more information about the instructions, see *OpenText Documentum Content Management - Branch Office Caching Services Cloud Deployment Guide (EDCCA-ICD)*.

17. Before deploying Documentum Content Ingestion Service (DCIS) to use OpenText™ Content Aviator:

    - Obtain the client ID and the client secret from the OTDS admin.
    - Obtain the embedding service endpoint and the chat service endpoint.
    - Obtain the SaaS Proxy Service WebSocket endpoint and user credentials.

- Obtain the ActiveMQ user credential and URL.

18. Before deploying OpenText Information Archive that uses OTDS from the OpenText Documentum CM single Helm package, ensure that you include all the Bootstrap configurations in the `documentum/charts/otds/charts/otdsws/config/config.yml` file.

    For more information about deploying OpenText Information Archive and configuring the Bootstrapping template, see *OpenText Information Archive - Cloud Deployment Guide (EARCORE250400-ICD)*.

19. Disable TLS 1.0, 1.1, and weaker ciphers in TLS 1.2 at the infrastructure level. Enable only the following ciphers for TLS 1.2:

    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

20. If you want to use OpenText Documentum CM with supported vault types, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)* and “Vault” on page 155.

2.2 Configuring Microsoft Azure cloud platform

1. Make sure that you complete all the relevant tasks described in “Downloading Helm and Helm charts, container images, and configuring cloud platform-specific requirements” on page 7.

2. To set up Azure in the Kubernetes cluster, do the following:

    a. Configure an Azure account.
    b. Create a resource group. A resource group in Azure is a folder to keep your collection. It does not serve any other purpose.
    c. Create a container registry. Container registry is used to store the container images in Azure. A standard container registry can store up to 100 GB of images.
    d. Create an Azure Kubernetes Service. Azure Kubernetes Service is a managed container orchestration service, based on the open source Kubernetes system, which is available on the Azure public cloud.
    e. Create an Azure database for the PostgreSQL server to enable Postgres as a service.
    f. Connect to Azure cluster using the Azure CLI command.
    g. Run the following command to create a storage account:

    ```
    az storage account create --resource-group <name of the resource group>
    --name <name of the storage account>
    --sku <SKU of the storage account>
    ```

    Example 2-1: Creating a storage class output

    ```
    [root@skvoraclelinux ~]# az storage account create
    --resource-group MC_dctm_dctmaks_eastus
    ```

```
name dctmstorageacc --sku Standard_LRS
{
"accessTier": null,
"creationTime": "2018-11-26T06:11:58.380457+00:00",
"customDomain": null,
"enableHttpsTrafficOnly": false,
"encryption": {
"keySource": "Microsoft.Storage",
"keyVaultProperties": null,
"services": {
"blob": {
...
...
[root@skvoraclelinux ~]#
```

h. Create a storage class, a YAML file (for example, `azstorageclass.yaml` with appropriate values for the parameters), and apply the configuration.

   A storage class is used to define how an Azure file share is created. A storage account can be specified in the class.

   Different types of storage are:

   - Locally-redundant storage (LRS): A simple, low-cost replication strategy. Data is replicated within a single storage scale unit.
   - Zone-redundant storage (ZRS): Replication for high availability and durability. Data is replicated synchronously across three availability zones.
   - Geo-redundant storage (GRS): Cross-regional replication to protect against region-wide unavailability.
   - Read-access geo-redundant storage (RA-GRS): Cross-regional replication with read access to the replica.

   To create a storage class, do the following:

   i. Run the following command to create a storage class:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: azurefile
provisioner: kubernetes.io/azure-file
mountOptions:
  - dir_mode=0777 #refers to permission mode
  - file_mode=0777
  - uid=1000 #refers to the user id of the OpenText Documentum CM
installation owner
  - gid=1000
parameters:
skuName: Standard_LRS
storageAccount: <name of the created storage account>
```

   ii. Run the following command to apply the configuration to a resource using the name of the YAML file:

```
kubectl apply -f <name of the YAML file>.yaml
```

Example 2-2: Applying the configuration output

```
[root@skvoraclelinux ~]# kubectl apply -f azstorageclass.yaml
storageclass.storage.k8s.io/azurefile created
```

Resource is created if it does not exist yet. Make sure that you specify the resource name.

i. Create and verify a successful sample persistent volume claim (PVC) creation with the newly created storage class (azurefile) in the ReadWriteMany access mode.

j. Download the container image(s) from OpenText Container Registry. See step 8.

k. Run the following command to tag the container image(s) to the Azure registry specific image:

```
docker tag <source image> <destination image>
```

l. Run the following command to sign in to the Azure container registries, if not already logged in:

```
az acr login --name <azure container registry>
```

m. Run the following command to upload the container image to the Azure Container Registry:

```
docker push <image>
```

n. Install the NGINX Ingress Controller.

   NGINX documentation contains detailed information.

Microsoft Azure documentation contains detailed information.

3. To set up Azure Red Hat OpenShift cluster, do the following:

   a. Configure an Azure account.
   b. Create a resource group. A resource group in Azure is a folder to keep your collection. It does not serve any other purpose.
   c. Create a Domain Name System (DNS) zone and configure a domain name.
   d. Create and configure a service principal.
   e. Create an Azure database for the PostgreSQL server to enable Postgres as a service.
   f. Create an Azure Red Hat OpenShift cluster.
   g. Run the following command to create a project:

```
oc new-project <project-name>
```

   h. Create storage class to support the ReadWriteMany access mode. For information to create a storage class to support the ReadWriteMany access mode, see *Red Hat* documentation.

      An example YAML to create a ReadWriteMany storage class:

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
```

```
metadata:
  name: azure-file
provisioner: kubernetes.io/azure-file
mountOptions:
  - dir_mode=0777
  - file_mode=0777
  - uid=1000
  - gid=1000
  - mfsymlinks
  - actimeo=30
  - nosharesock
parameters:
  location: centralus
  secretNamespace: kube-system
  skuName: Standard_LRS
  storageAccount: dctmazurestorageaccount
  resourceGroup: azurefile_rg
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

i. Run the following command to create the storage class:

```
oc create -f <name of the YAML file>.yaml
```

ii. Make sure that there is a storage class to support both the ReadWriteOnce and ReadWriteMany access modes before you proceed with the deployment.

i. When a user logs in to Red Hat OpenShift, by default, the user is added to the restricted security context constraints (SCC). For some privilege operation such as `chmod`, you must provide specific privileges to the pods as per the environment and security requirements. For example, perform the following steps:

   i. (Recommended) Run the following command to add the default service account deployer to `anyuid`:

   ```
   oc adm policy add-scc-to-user anyuid -z default
   ```

   ii. Run the following command to add all authenticated users to `anyuid scc` instead of `restricted`:

   ```
   oc adm policy add-scc-to-group anyuid system:authenticated
   ```

   iii. If the OTDS deployment fails due to a seccomp error or an error caused by the `uid` value of 1000, perform the following steps:

      A. Run the following command to create a custom SCC, for example, `allow-uid-1000`:

      ```
      oc create -f <custom scc>.yaml
      ```

      Example 2-3: Custom SCC YAML file

      ```
      apiVersion: security.openshift.io/v1
      kind: SecurityContextConstraints
      metadata:
        name: allow-uid-1000
      allowPrivilegeEscalation: false
      allowPrivilegedContainer: false
      allowHostDirVolumePlugin: false
      allowHostIPC: false
      allowHostNetwork: false
      ```

```
allowHostPID: false
allowHostPorts: false
allowLocalVolumes: false
allowSupplementalGroups: true
allowedCapabilities: []
allowedFlexVolumes: []
allowedUnsafeSysctls: []
defaultAddCapabilities: []
fsGroup:
  type: MustRunAs
  ranges:
    - min: 1000
      max: 1000
readOnlyRootFilesystem: false
requiredDropCapabilities:
  - KILL
  - MKNOD
  - SETGID
  - SETUID
runAsUser:
  type: MustRunAs
  uid: 1000
seLinuxContext:
  type: RunAsAny # Allows any SELinux context
seccomp:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
volumes:
  - configMap
  - downwardAPI
  - emptyDir
  - persistentVolumeClaim
  - projected
  - secret
  - serviceAccountToken
priority: 10
```

B. Run the following command to create a service account, for example, `otds-sa`:

```
kubectl create sa <service-account-name>
```

C. Run the following command to assign the service account to the custom SCC:

```
oc patch <custom scc> --type=json -p='[{"op": "add", "path": "/
users/-", "value": "system:serviceaccount:<namespace>:<service-
account-name>"}]' echo "Patch command exit code: $?"
```

In the `documentum/values.yaml` file, under the `otds` section, set this service account as the value of the `serviceAccountName` variable.

D. Open the `otds-deployment.yaml` file in the `documentum\charts\otds\charts\otdsws\templates` folder and remove the following lines from the `securityContext` section:

```
runAsUser: {{ .Values.securityContext.runAsUser }}
{{- if semverCompare
">=1.19-0" .Capabilities.KubeVersion.GitVersion }}
seccompProfile:
  type: RuntimeDefault
```

j. Create a container registry in Azure.

k. Download the container image from OpenText Container Registry. See step 8.

l. Run the following command to tag the container image(s) to the Azure registry specific image:

```
docker tag <source image> <destination image>
```

m. Run the following command to sign in to the Azure container registries, if not already logged in:

```
az acr login --name <azure container registry>
```

n. Run the following command to upload the container image to the Azure Container Registry:

```
docker push <image>
```

o. Make sure that images are accessible in the Azure Red Hat OpenShift cluster from Azure container registry.

p. Make sure that the Ingress Controller is up and running.

q. Update the ingress annotations in the `documentum/platforms/openshift.yaml` file as per the environment requirement.

r. In the `documentum/config/passwords.yaml` file, under the `ingress` section, set the unencrypted certificate and key as the value of the `tlscrt` and `tlskey` variables, instead of the Base64-encoded TLS certificate and key as the OpenShift routes do not support password-protected private keys.

For example:

```
Ingress:
  tlscrt:|
    -----BEGIN CERTIFICATE-----
    MIIDtzCCAzygAwIBAgISBbIGRyU7PHyI1Ubxl05ypKB2MAoGCCqGSM49BAMDMDIx
    CzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MQswCQYDVQQDEwJF
    NjAeFw0yNTA0MDMxMTUyNTlaFw0yNTA3MDIxMTUyNThaMB4xHDAaBgNVBAMMEyou
    ZG9jdW1lbnR1bS1xZS5uZXQwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAATDv0Y1
     NFuwaEiX2gFAhO8iO3OawDv/vQufkTzCmbmti8DOA3XD2DGbzS7IMS8WrCCjt2OT
    U/Tlybi5tlmUAL+8o4ICRDCCAkAwDgYDVR0PAQH/BAQDAgeAMB0GA1UdJQQWMBQG
    CCsGAQUFBwMBBggrBgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1UdDgQWBBQokFOS
    scTduLmmUApcGWosxvIdKzAfBgNVHSMEGDAWgBSTJ0aYA6lRaI6Y1sRCSNsjv1iU
    OjBVBggrBgEFBQcBAQRJMEcwIQYIKwYBBQUHMAGGFWh0dHA6Ly9lNi5vLmxlbmNy
    Lm9yZzAiBggrBgEFBQcwAoYWaHR0cDovL2U2LmkubGVuY3Iub3JnLzAeBgNVHREE
    FzAVghMqLmRvY3VtZW50dW0tcWUubmV0MBMGA1UdIAQMMAowCAYGZ4EMAQIBMC4G
    A1UdHwQnMCUwI6AhoB+GHWh0dHA6Ly9lNi5jLmxlbmNyLm9yZy8xMTkuY3JsMIIB
    AwYKKwYBBAHWeQIEAgSB9ASB8QDvAHUA3dzKNJXX4RYF55Uy+sef+D0cUN/bADoU
    EnYKLKy7yCoAAAGV+7TgKQAABAMARjBEAiBoZzYepSr+GqQKYucmlzP3dD+O6P5m
    UY1UnKt7sslC8gIgA2ytbdEBmNANArYogkjkOrMPN6CNYvGXYdm7glTQHdwAdgAN
    4fIwK9MNwUBiEgnqVS78R3R8sdfpMO8OQh60fk6qNAAAAZX7tOfPAAAEAwBHMEUC
    IEwnvzBqwrGmqWJUunqFwawQT8MayBJDDL6ugw354ZrEAiEA2oqUBKZkLvq8Hubx
    58gJEyo5S+BYRULfMlWNCYO0AYAwCgYIKoZIzj0EAwMDaQAwZgIxAPV8EAH5KyIA
    ZX1fxm8m2n/BqW9eAYTwByzxezeWs8e/HHRNAKTbtM21J1DbJghwnQIxAJNQd4qc
    1ne3U4ZUfDsjA9k7iPZIBMwTsWP8Gl7+eeRDygEmQpDG4GrTcVWbtAGdsQ==
    -----END CERTIFICATE-----
  tlskey:|
    -----BEGIN PRIVATE KEY-----
    MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgsrsUW69Ryiw6ehi+
    BaWiqGraIKEJK5pxspGN8xQ49ZehRANCAATDv0Y1NFuwaEiX2gFAhO8iO3OawDv/
    vQufkTzCmbmti8DOA3XD2DGbzS7IMS8WrCCjt2OTU/Tlybi5tlmUAL+8
    -----END PRIVATE KEY-----
```

Microsoft Azure and *Red Hat OpenShift* documentation contain detailed information.

4. To set up Azure in the Red Hat OpenShift cluster on bare metal, do the following:

   a. Make sure that you have a Linux VM (with the supported CentOS operating system) with hard disk more than 100 GB, RAM 32 GB and 8 CPUs and add a user who is part of the wheel group.
   b. Download CodeReady Containers (CRC) from the Red Hat website.
   c. Install Network Manager.
   d. Include the CodeReady Containers path in the <*PATH*> environment variable.
   e. Fetch the `pull secret` configuration file from the Red Hat account.
   f. Run the following commands:

      ```
      crc setup
      crc start -p <pull_secret_file>
      ```
   g. After the cluster is ready, run the following command to sign in as `kubeadmin`:

      ```
      oc login -u kubeadmin -p <password>
      ```
   h. Run the following command to configure Security Context Constraints (SCCs):

      ```
      oc adm policy add-scc-to-user anyuid -z default
      oc adm policy add-scc-to-group anyuid system:authenticated
      ```
   i. Run the following command to create a project:

      ```
      oc adm new-project <project_name>
      ```
   j. Run the following command to switch to the new project:

      ```
      oc project <project_name>
      ```
   k. Make sure that there is a storage class to support both the ReadWriteOnce and ReadWriteMany access modes before you proceed with the deployment.
   l. Download the container image (Oracle Linux only) from OpenText Container Registry. See step 8.
   m. Run the following command to upload the container image to the bare metal OpenShift container registry:

      ```
      docker push <image>
      ```

5. Optional To enable Documentum CM Server/connection broker external access, update the values for the following variables:

   a. Set the value of `externalAccessEnaled` to `true` in `common-variables`.
   b. Retain all the default values for `ExtDocbroker` in `docbroker`.
   c. Set the value of `ExtCS.nativeExtPort` to `80` and `sslExtPort` to `81` in `content-server`. Retain the default values for all other variables.

   When the Documentum CM Server component is enabled, Ingress is enabled by default.

6. Set the value of `dctm-ingress.ingress.configureHost` to `false` in the `documentum/config/configuration.yml` file.

   Note: When you set the value of `dctm-ingress.ingress.configureHost` to `true`, you must provide the appropriate values for `global.hostShortName` and `global.ingressDomain` in the `documentum/values.yaml` file.

7. Optional If you performed the tasks in step 5, obtain the external IP address of the `csext<sname>` load balancer service using the `kubectl get svc` command, and then change the default value of `ExtCS.tcp_route` to the new external IP address.

8. Run the Helm upgrade command.

9. To access Documentum CM Server by external clients (outside the cluster), Documentum CM Server and connection broker must be deployed with externalization configuration as mentioned in step 5, step 7, and step 8. After the successful deployment, two load balancer services are created, one each for `csext-<sname>` and `dbrext-<sname>` having their own external IP address.

   Copy the `dfc.properties` file from `/opt/dctm/config/` to your client machine from the primary Documentum CM Server pod. The original `dfc.properties` file contains two sets of host and port pair. Remove one set of host and port pair and specify the following for the other set:

   ```
   dfc.docbroker.host[0]=<External IP of dbrext load balancer service>
   dfc.docbroker.port[0]=<ExtCS.nativeExtPort>
   ```

   For example:

   ```
   dfc.docbroker.host[0]=10.70.62.110
   dfc.docbroker.port[0]=80
   ```

2.3 Configuring Amazon Web Services cloud platform

1. Make sure that you complete all the relevant tasks as described in "Downloading Helm and Helm charts, container images, and configuring cloud platform-specific requirements" on page 7.

2. To set up Amazon Web Services in the Amazon Elastic Kubernetes Service (EKS) Amazon Elastic Compute Cloud (EC2) cluster, do the following:

   a. Create resources on Amazon Web Services. You must be an Identity and Access Management (IAM) user with the following roles and permissions:

      - Roles: EKS to allow EKS to manage clusters and EC2 to allow instances to call Amazon Web Services services.
      - Permissions: `elasticfilesystem:CreateFileSystem` and `elasticfilesystem:CreateMountTarget`.

   b. Create an EKS cluster and a namespace within the cluster.

i. Install and configure Amazon Web Services CLI.

ii. Run the following command to verify the `kubectl` installation:

```
kubectl version --short --client
```

Note: You must use a kubectl version that is within one minor version difference of your Amazon EKS cluster control plane. For example, a 1.20 kubectl client must work with Kubernetes 1.19, 1.20, and 1.21 clusters.

The product *Release Notes* contains the supported versions of the Kubernetes cluster.

iii. Create an EKS Cluster.

iv. Run the following command to create a Kubernetes cluster namespace on the Cloud shell:

```
kubectl create namespace <name of namespace>
```

v. Run the following command to verify if the namespace is created:

```
kubectl get namespace
```

c. Run the following command to authenticate Amazon Elastic Container Registry (ECR) from CLI:

```
aws ecr get-login-password --region <region> | docker login --username AWS --
password-stdin <AWS account ID>.dkr.ecr.<region>.amazonaws.com
```

Example 2-4: Authenticating Amazon ECR from CLI

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --
password-stdin 777777777.dkr.ecr.us-west-2.amazonaws.com
```

d. Create a repository in ECR to load your container images.

e. Download the container image(s) from OpenText Container Registry. See step 8.

f. To upload the container image(s) to Amazon ECR, do the following:

i. Run the following command to tag your container image(s):

```
[aws_account_id.dkr.ecr.region.amazonaws.com]/[name of repository]/[image
tag]
```

ii. Run the following command to upload the tagged container image(s) to ECR:

```
docker push 777777777.dkr.ecr.region.amazonaws.com/
dctmrepository:<image_tag>
```

g. Extract the Helm package you downloaded described in “Downloading Helm and Helm charts, container images, and configuring cloud platform-specific requirements” on page 7 to a temporary location.

h. Install the supported version of the Helm package.

The product *Release Notes* contains detailed information about the supported version.

i. Create an Amazon Elastic File System (EFS) file system.

j. Deploy the Amazon EFS Container Storage Interface (CSI) driver.

Notes

- Amazon EFS CSI driver dynamic provisioning is supported from the 23.2 release.
- If you are deploying Documentum CM Server 23.2 (new deployment), OpenText recommends you to use dynamic provisioning.
- If you are upgrading from a previous deployment of Documentum CM Server versions (for example, 25.2) that used static provisioning, then ensure that you use the same static provisioning storage class with the same Helm inputs (for example, `existVolumePv`, `awsEFS`, `awsEFSCSIDriver`, and `awsEFSCSIHandle`) used in the previous deployment (for example, 25.2) while upgrading to 25.4.

k. Create a storage class (for example, `efs-sc`) using CSI driver and test the sample PV and PVC bindings for both the Read Write Once and Read Write Many access modes.

Note: Skip this note about creating additional storage class if you are using Amazon Aurora or PostgreSQL database service.

- When you are installing the EFS CSI driver with dynamic provisioning using the *Amazon Web Services* documentation, make sure to change the following variables in addition to `fileSystemId` in the `storageclass.yaml` file before creating the storage class as follows:

  From:

  ```
  gidRangeStart: "1000" # optional
  gidRangeEnd: "2000" # optional
  ```

  To:

  ```
  uid: "1000" # optional
  gid: "1000" # optional
  ```

- If you are using the Postgres container for database which uses the Postgres image internally, then make sure to create another storage class (for example, `efs-dbpg`) with `uid` and `gid` values as follows:

  ```
  uid: "999" # optional
  gid: "999" # optional
  ```

  Use this (`efs-dbpg`) storage class in `persistentVolume.storageClass` in the `documentum/charts/db/values.yaml` file.

l. Delete both the sample PV and PVC resources. Make sure that you do not delete the storage class (for example, `efs-sc`).

m. Optional Amazon Aurora or PostgreSQL database service are supported. Make sure that you set the required security settings (for example, Virtual

Private Cloud security groups) in the database instance to access it from the EKS cluster deployments.

i. When the database instance is created successfully and is up and running, do the following:

   A. Navigate to the Amazon Relational Database Service (RDS) console.
   B. Select the database instance you created.
   C. Record the name provided for **Endpoint** and the **Database instance identifier** information.

ii. To use Amazon Aurora or PostgreSQL database instance with the Helm deployment, do the following:

   A. Open the `documentum/values.yaml` file and provide the endpoint information from step 2.m.i.C for `global.db_hostname` and the port number for `global.dbport`.
   B. Open the `documentum/config/configuration.yml` file and provide the database instance identifier information from step 2.m.i.C for `global.db_service`.
   C. Open the `documentum/config/passwords.yaml` or `documentum/config/passwords_vault.yaml` (if Hashicorp Vault is enabled) or `documentum/config/passwords_k8api.yaml` file (if Kubernetes native secrets is enabled) and provide the appropriate value for `databaseAdminPassword`.
   D. Open the `documentum/config/configuration.yml` file and provide the appropriate value for `cs-secrets.database.userName`.

Amazon Web Services documentation contains detailed information to set up Amazon RDS, create Amazon Aurora or PostgreSQL database instance, and to connect to the database.

Note: If the database is enabled with Secure Sockets Layer (SSL), update the following variables:

- Open the `documentum/values.yaml` file and set the value of `global.db_ssl` to `true` and set the value of `global.db_ssl_mode` to the required SSL connection mode.

- Open the `documentum/config/passwords.yaml` or `documentum/config/passwords_vault.yaml` (if Hashicorp Vault is enabled) or `documentum/config/passwords_k8api.yaml` (if Kubernetes native secrets is enabled) and provide the appropriate value for `cs-secrets.database.certificate`.

n. Deploy the Amazon Web Services Application Load Balancer (ALB) Ingress Controller add-on Helm to your EKS cluster.

o. Run the following command to verify the status of the Amazon Web Services ALB Ingress Controller Helm deployment:

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

Amazon Web Services documentation contains detailed information.

3. To set up Amazon Web Services in the Amazon Elastic Kubernetes Service (EKS) Fargate cluster, do the following:

   a. Set up the Amazon Web Services EKS Fargate environment.

   b. Create a dedicated namespace in the Amazon EKS Fargate environment. Make sure that any pod that is deployed in this environment are served by Fargate nodes or profiles.

   c. Perform all the relevant steps described in step 2.

   d. Create the required PV and PVCs manually for the pod deployment as Amazon Web Services cloud platform supports only Amazon Web Services EFS CSI static provisioning for the Amazon Web Services EKS Fargate environment.

      Make sure that unique access points are created for each PV as follows:

      - **User ID**, **Group ID**, **Owner user ID**, and **Owner group ID** set to `1000`.
      - Permission set to `777`.
      - Unique root folder path for each access point.

Example 2-5: YAML file to create PV and PVC

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: <pv-name>
spec:
  capacity:
    storage: <size>
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  storageClassName: <storage-class-name>
  csi:
    driver: efs.csi.aws.com
    volumeHandle: <file system ID>::<access point ID>
  persistentVolumeReclaimPolicy: Delete
  volumeMode: Filesystem
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <pvc-name>
  namespace: <namespace>
  labels: app.kubernetes.io/managed-by: Helm
  annotations:
    meta.helm.sh/release-name: <release-name>
    meta.helm.sh/release-namespace: <namespace>
spec:
  accessModes:
  - <access-mode>
  storageClassName: <storage-class-name>
  volumeMode: Filesystem
  volumeName: <pv-name>
  resources:
    requests:
      storage: <size>
```

e. Update the image and tag fields in the `documentum/dockerimages-values.yaml` file to point to the ECR images.

f. Amazon Web Services Fargate is supported if you have enabled the components in the `documentum/documentum-components.yaml` file of the following container images:

- `otds`
- `docbroker`
- `content-server`
- `cs-secrets`
- `dctm-dms`
- `contentconnect`
- `dctm-rest`
- `dfs`
- `dctm-cmis`
- `dctm-workflow-designer`
- `bps`
- `xda`
- `da`
- `records`
- `rqm`
- `dtrbase`
- `xPlore-OneD`
- `dctmclientinstaller`
- `peinstaller`
- `d2classic`
- `d2config`
- `d2rest`
- `d2smartview`
- `adminconsole`
- `smartviewm365`
- `oesconnector`
- `notificationservice`

- `dctm-dcc`
- `otiv`
- `sapconnector-init`
- `dctm-assap`
- `dctm-assap-ilm`
- `dctm-cssap`

To deploy in a non-vault and non-SSL mode, you must create PVs and PVCs as described in the following table:

PVC name	PV name	App	Access mode	Storage	PV reclaim policy
adminconsole-vct-adminconsole-0	fargatepv1	adminconsole	ReadWrite Once	lGi	Delete
adminconsole-vct-adminconsole-1	fargatepv2	adminconsole	ReadWrite Once	lGi	Delete
shared-logs-adminconsole-0	fargatepv3	adminconsole	ReadWrite Once	2Gi	Delete
shared-logs-adminconsole-1	fargatepv4	adminconsole	ReadWrite Once	2Gi	Delete
bps-vct-bps-0	fargatepv5	bps	ReadWrite Once	1Gi	Delete
custom-script-pvc	fargatepv6	dctm-rest custom scriptPVCname	ReadWrite Many	1Gi	Delete
cmis-pvc	fargatepv7	cmis	ReadWrite Many	2Gi	Delete
dctm-cmis-logs-pvc	fargatepv8	cmis	ReadWrite Many	2Gi	Delete
da-pvc	fargatepv9	da	ReadWrite Once	lGi	Delete
dbr-vct-dbr-0	fargatepv10	dbr	ReadWrite Once	1Gi	Delete
shared-logs-dbr-0	fargatepv11	dbr	ReadWrite Once	2Gi	Delete

PVC name	PV name	App	Access mode	Storage	PV reclaim policy
shared-logs-dbr-1	fargatepv12	dbr	ReadWrite Once	2Gi	Delete
dbr-vct-dbr-1	fargatepv13	dbr	ReadWrite Once	1Gi	Delete
dcs-vct-dcs-pg-0	fargatepv14	dcs-pg	ReadWrite Once	1Gi	Delete
dcs-vct-dcs-pg-1	fargatepv15	dcs-pg	ReadWrite Once	1Gi	Delete
shared-logs-dcs-pg-0	fargatepv16	dcs-pg	ReadWrite Once	2Gi	Delete
dcs-pg-pvc	fargatepv17	dcs-pg	ReadWrite Many	1Gi	Retain
shared-logs-dcs-pg-1	fargatepv18	dcs-pg	ReadWrite Once	2Gi	Delete
dctm-worflow-designer-pvc	fargatepv19	dctm-worflow-designer	ReadWrite Once	2Gi	Delete
d2classic-vct-d2classic-0	fargatepv20	d2classic	ReadWrite Once	lGi	Delete
shared-logs-d2classic-0	fargatepv21	d2classic	ReadWrite Once	2Gi	Delete
d2config-vct-d2config-0	fargatepv22	d2config	ReadWrite Once	1Gi	Delete
d2custom-shared-pvc	fargatepv23	d2config	ReadWrite Many	2Gi	Delete
shared-logs-d2config-0	fargatepv24	d2config	ReadWrite Once	2Gi	Delete
shared-logs-d2rest-0	fargatepv25	d2rest	ReadWrite Once	2Gi	Delete
d2rest-vct-d2rest-0	fargatepv26	d2rest	ReadWrite Once	lGi	Delete
d2smartview-vct-d2smartview-0	fargatepv27	d2smartview	ReadWrite Once	1Gi	Delete
shared-logs-d2smartview-1	fargatepv28	d2-smartview	ReadWrite Once	2Gi	Delete

PVC name	PV name	App	Access mode	Storage	PV reclaim policy
d2smartview-vct-d2smartview-1	fargatepv29	d2smartview	ReadWrite Once	1Gi	Delete
shared-logs-d2smartview-0	fargatepv30	d2smartview	ReadWrite Once	2Gi	Delete
d2-extension-pvc	fargatepv31	d2 common PVC	ReadWrite Many	8Gi	Delete
dfs-server-dfs-extension-pvc	fargatepv32	dfs	ReadWrite Once	2Gi	Retain
dfs-server-vct-dfs-server-0	fargatepv33	dfs	ReadWrite Once	1Gi	Delete
dctm-dms-logs-pvc	fargatepv34	dms	ReadWrite Many	2Gi	Delete
dms-pvc	fargatepv35	dms	ReadWrite Many	1Gi	Delete
ns-logs-pvc	fargatepv36	notification service	ReadWrite Many	4Gi	Delete
oesconnector-logs-pvc	fargatepv37	oesconnector	ReadWrite Many	4Gi	Delete
data-otiv-amqp-0	fargatepv38	otiv	ReadWrite Once	1Gi	Delete
data-otiv-amqp-1	fargatepv39	otiv	ReadWrite Once	1Gi	Delete
otiv-bas	fargatepv40	otiv	ReadWrite Once	2Gi	Delete
dtr-pvc	fargatepv41	reports	ReadWrite Many	1Gi	Delete
records-ext-conf-pvc	fargatepv42	records	ReadWrite Many	1Gi	Delete
rqm-pvc	fargatepv43	rqm	ReadWrite Many	2Gi	Delete
assap-custom-pvc	fargatepv44	sapconnector	ReadWrite Many	22Mi	Delete
assap-ilm-custom-pvc	fargatepv45	sapconnector	ReadWrite Many	22Mi	Delete

PVC name	PV name	App	Access mode	Storage	PV reclaim policy
assap-ilm-pvc	fargatepv46	sapconnector	ReadWrite Many	1Gi	Delete
assap-pvc	fargatepv47	sapconnector	ReadWrite Many	1Gi	Delete
cssap-pvc	fargatepv48	sapconnector	ReadWrite Many	1Gi	Delete
cssap-custom-pvc	fargatepv49	sapconnector	ReadWrite Many	22Mi	Delete
smartview365-logs-pvc	fargatepv50	smartview365	ReadWrite Once	4Gi	Delete
xda-pvc	fargatepv51	xda	ReadWrite Many	1Gi	Delete
xpl-data-pvc	fargatepv52	xPlore	ReadWrite Many	29Gi	Retain
xdbbackup	fargatepv53	xPlore	ReadWrite Many	1Gi	Retain
d2-extension-pvc-da	fargatepv54	da	ReadWrite Many	2Gi	Delete
d2-extension-pvc-rm	fargatepv55	rm	ReadWrite Many	2Gi	Delete

Notes

- After enabling OpenText Intelligent Viewing, do the following:
  - In the `documentum\password.yaml` file, ensure that the value of the `otiv.global.newRelic.licenseKey` variable is blank as New Relic must be disabled for OpenText Intelligent Viewing.
  - If there is any issue with the pods, review and increase the resource requests and limits (CPU and memory) to ensure sufficient resources for stable operation.
- If the value of `otiv.global.transformationPvcRWO` variable is true, change the access mode of `otiv-bas` to `ReadWriteOnce`.

  When you enable OpenText Intelligent Viewing, `otiv-bas` PVC is created by the Helm. Create the PV and include the following additional parameter:

  ```
  claimRef:
    namespace: <namespace>
    name: otiv-bas
  ```
- All the non-certified components must remain `disabled` to maintain platform compatibility and ensure successful deployment.

- You can modify the PVC storage size according to your requirement.
- If you want to increase the replica count of any pod after the successful deployment of pods, make sure that you create corresponding PVs and PVCs appropriately according to your requirement before scaling up the deployment of the same pod.
- If the value of `global.tomcatbase_usecommonpvc` in the `documentum/values.yaml` file is set to `true`, creation of `dfs-server-dfs-extension-pvc`, `cmis-pvc`, and `dms-pvc` are not required.
- If the value of `global.tomcatbase_usecommonpvc` in the `documentum/values.yaml` file is set to `false`, creation of `d2-extension-pvc-da` and `d2-extension-pvc-rm` are required.

g. After you create the PVs and PVCs, make sure that the `STATUS` of all PVs and PVCs are `Bound`.

h. Make sure that the values of `resources.limits` and `resources.requests` in `content-server` in `<location where Helm charts are extracted>/documentum-resources-values-<config>.yaml` are set as per your sizing and Fargate environment requirement where `<config>` can be `extra-large`, `large`, `medium`, `medium-large`, `small`, or `small-medium` resource value YAML files.

   The available resource YAML files contain pod sizing values, including CPU and memory.

i. Make sure that the values of `resources.limits` and `resources.requests` in `content-server` are set as per your sizing or environment requirement.

Amazon Web Services documentation contains detailed information.

4. To set up Red Hat OpenShift cluster on Amazon Web Services (ROSA), do the following:

   a. Configure the Amazon Web Services and Red Hat account.

   b. Install Amazon Web Services CLI, ROSA CLI, and OpenShift CLI.

      These tools are required to deploy, configure, and manage the OpenShift cluster on Amazon Web Services.

   c. Create the Amazon Web Services user with the required permissions. For more information, see *Red Hat Documentation*.

   d. Configure your local machine with the new Amazon Web Services user credentials using the Amazon Web Services CLI.

      This ensures secure authentication and allows you to install and manage the Red Hat OpenShift cluster on Amazon Web Services cluster seamlessly using the ROSA CLI or Amazon Web Services CLI commands.

   e. Enable ROSA in Amazon Web Services.

   f. Run the following command to create the Red Hat OpenShift cluster:

```
rosa create cluster --cluster-name <cluster_name>  --sts --mode auto
```

While installing the cluster, run the following command to monitor the logs:

```
rosa logs install -c <cluster-name> --watch
```

Copy the login command from the log file to log in to your cluster.

For example:

```
oc login https://api.openshift1.mas3.p1.openshiftapps.com:6443 --username
cluster-admin --password LTPGQ-j4R6R-9W4yf-FNRCy
```

g. Install EFS Drivers for storage classes. For more information, see *Red Hat Documentation*.

h. Create the Amazon EFS file system in the Amazon Web Services Management Console.

   This file system is used to create a storage class in the Red Hat OpenShift cluster for persistent storage.

i. Create a storage class to support the ReadWriteMany access mode. For information about creating a storage class to support the ReadWriteMany access mode, see *Red Hat documentation* .

Example 2-6: YAML to create a ReadWriteMany storage class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage class name>
  annotations:
    description: 'openshift storage class'
provisioner: efs.csi.aws.com
parameters:
  basePath: /dynamic_provisioning
  directoryPerms: '700'
  fileSystemId: <newly created filesystem Id>
  gid: '1000'
  provisioningMode: efs-ap
  uid: '1000'
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

   i. Run the following command to create the storage class:

```
oc create -f <name of the YAML file>.yaml
```

   ii. Ensure that there is a storage class to support both the ReadWriteOnce and ReadWriteMany access modes before you proceed with the deployment.

j. Use the existing OpenShift container registry in Amazon Web Services and perform the following steps to pull images to OpenShift container registry:

   i. Run the following command to pull images to local machine using docker:

```
docker pull <repository>.<imagename>:<imagetag>
```

ii. Run the following command to check the registry hostname:

```
oc get route default-route -n openshift-image-registry
```

iii. Run the following command to retag the image using the OpenShift registry hostname:

```
Docker tag <repository>.<imagename>:<imagetag> < registry-host
>.<imagename>:<imagetag>
```

iv. Run the following command to login to the OpenShift container registry:

```
docker login -u <username> -p <login-token> <registry-host-url>
```

v. Run the following command to provide access permissions to the namespace containing the pod that can pull images from the OpenShift container registry:

```
oc policy add-role-to-user system:image-puller
system:serviceaccount:<namespace>:<service-account> --namespace=<container
registry namespace> (default "cs")
oc policy add-role-to-user system:image-builder
system:serviceaccount:<namespace>:<service-account> --namespace=<container
registry namespace> (default "cs")
```

vi. Run the following command to push the image to your cluster's container registry:

```
docker push <registry-host>/<project-name>/<image-name>:<tag>
```

k. When a user logs in to Red Hat OpenShift, by default, the user is added to the restricted security context constraints (SCC). The OTDS and OpenText™ Intelligent Viewing pods require `uid` value of 1000, which is outside the allowed range of the default SCC. To run the OTDS and OpenText Intelligent Viewing pods, perform the following steps:

i. Run the following command to create a service account for OTDS:

```
kubectl create sa <service-account-name>
```

Note: OpenText Intelligent Viewing creates the service account during the pre-hook upgrade.

ii. Run the following command to create a custom SCC:

```
oc create -f <custom scc>.yaml
```

Example 2-7: Custom SCC YAML file

```
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
 name: otdsws-anyuid-scc
priority:10
allowHostDirVolumePlugin: false
allowHostIPC: false
allowHostNetwork: false
allowHostPID: false
allowHostPorts: false
allowPrivilegeEscalation: true
allowPrivilegedContainer: false
```

```
allowedCapabilities: null
defaultAddCapabilities: []
fsGroup:
  type: RunAsAny
groups: []
readOnlyRootFilesystem: false
requiredDropCapabilities: []
runAsUser:
  type: MustRunAs
  uid: 1000
seLinuxContext:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
users:
  - system:serviceaccount:<namespace:serviceaccount>
volumes:
 - awsElasticBlockStore
 - azureDisk
 - azureFile
 - cephFS
 - cinder
 - configMap
 - csi
 - downwardAPI
 - emptyDir
 - ephemeral
 - fc
 - flexVolume
 - flocker
 - gcePersistentDisk
 - gitRepo
 - glusterfs
 - image
 - iscsi
 - nfs
 - persistentVolumeClaim
 - photonPersistentDisk
 - portworxVolume
 - projected
 - quobyte
 - rbd
 - scaleIO
 - secret
 - storageOS
 - vsphere
```

iii. To assign the service account to the custom SCC, perform the following steps:

A. Run the following command to edit the custom SCC:

```
oc edit scc <custom scc>
```

For example:

```
oc edit scc otdsws-anyuid-scc
```

B. Add the service account under the `users` section in the custom SCC:

```
users:
- system:serviceaccount:<namespace:serviceaccount>
```

For example:

```
users:
- system:serviceaccount:testuser:otdsws-sa
- system:serviceaccount:testuser:otiv-delete-amqp-sa
- system:serviceaccount:testuser:otiv-delete-secrets-sa
- system:serviceaccount:testuser:otiv-pvc-job
- system:serviceaccount:testuser:otiv-pvc-sa
- system:serviceaccount:testuser:stop-transformation-pods
- system:serviceaccount:testuser:otiv-deployment-sa
```

l. Run the following command to create a project in your cluster:

```
oc new-project <project-name>
```

Amazon Web Services and *Red Hat* documentation contains detailed information.

5. Set the value of `rwoStorage` and `rwmStorage` in the `documentum/values.yaml` file to the storage class name created in step 2.k using dynamic provisioning (for example, `efs-sc`).

Example 2-8: Setting the value of rwoStorage and rwmStorage

```
rwoStorage: &rwoStorage efs-sc
rwmstorage: &rwmStorage efs-sc
```

Note: If you are deploying Documentum CM Server in the Amazon Web Services EKS Fargate environment with EFS CSI static provisioning, make sure that you set the value of `persistentVolume.createPVC` to `false` in the `documentum/config/configuration.yml` file because PV and PVCs are already created manually as described in step 3.d.

When the Documentum CM Server component is enabled, Ingress is enabled by default.

6. Set the value of `dctm-ingress.ingress.configureHost` to `false` in the `documentum/config/configuration.yml` file.

Notes

- When you set the value of `dctm-ingress.ingress.configureHost` to `true`, you must provide the appropriate values for `global.hostShortName` and `global.ingresDomain` in the `documentum/values.yaml` file.
- Make sure that the value of `jmsBase` is set to `false` if you are using Amazon Web Services ALB as the Ingress Controller.

7. Open the `aws.yaml` file in the extracted `platforms` folder and perform the following step:

   In the `alb.ingress.kubernetes.io/subnets` annotation, provide the clusters' public subnet IDs separated by comma.

Example 2-9: Providing clusters public subnet ID

```
alb.ingress.kubernetes.io/subnets: subnet-0c4a1017a4ffb7962,
subnet-0bbd3ec1319a30772, subnet-0a6ab032a5e03be49
```

8. Optional To enable Documentum CM Server or connection broker external access, update the values for the following variables:

   a. Set the value of `externalAccessEnabled` to `true` in the `documentum/values.yaml` file.
   b. Retain all the default values for `docbroker.ExtDocbroker` and make sure to change the value of `ExtDocbroker.useELB` to `true` in the `documentum/config/configuration.yml` file.
   c. Set the value of `content-server.ExtCS.nativeExtPort` to `80`, `content-server.ExtCS.sslExtPort` to `81`, and `content-server.ExtCS.useELB` to `true` in the `documentum/config/configuration.yml` file. Retain the default values for all other variables.
   d. Documentum CM Server must be deployed with the externalization configuration using the preceding steps. Deploy the Documentum CM Server pod using the instructions provided in "Deploying OpenText Documentum CM using manual steps" on page 45. After the initial deployment of the pod, a load balancer service is created for `ExtCS` with an external IP address. Copy the external IP address and provide it as a value for `content-server.ExtCS.tcp_route`.

      For example:

      ```
      ExtCS:
          tcp_route:abc.aws.abc.com
      ```
   e. Run the Helm upgrade command.

Note: When you change the default value of any of the variables in `cs-logging-configMap` in the `documentum/values.yaml` file each time after the initial deployment, you must run the Helm upgrade command.

Amazon Web Services documentation contains detailed information.

2.4 Configuring Google Cloud Platform

1. Make sure that you complete all the relevant tasks described in "Downloading Helm and Helm charts, container images, and configuring cloud platform-specific requirements" on page 7.

2. Download and install the latest version of Google Cloud SDK from the Google Cloud Platform website on your machine which includes gcloud command line utility.

3. Create a Google Kubernetes Engine (GKE) and a namespace within the cluster using the following steps:

   a. Create a GKE cluster.
   b. Create namespace within the cluster.
   c. Create the NGNIX Ingress Controller in the GKE cluster.

4. Download the container image(s) from OpenText Container Registry. See step 8.

5. Upload the container image(s) to Google Container Registry (GCR) using the following steps:

   a. Run the following command to configure to use gcloud as a credential helper:

      ```
      gcloud auth configure-docker
      ```

   b. Run the following command to tag your container image(s):

      ```
      gcr.io/documentum-product/ot-dctm-server:<version>
      ```

   c. Run the following command to upload the tagged container image(s) to GCR:

      ```
      docker push gcr.io/documentum-product/ot-dctm-server:<version>
      ```

6. Make sure that there is a storage class to support both the ReadWriteOnce and ReadWriteMany access modes before you proceed with the deployment.

7. Optional To enable Documentum CM Server or connection broker external access, update the values for the following variables:

   a. Set the value of `externalAccessEnabled` to `true` in the `documentum/values.yaml` file.

   b. Retain all the default values for `docbroker.ExtDocbroker` in the `documentum/config/configuration.yml` file.

   c. Set the value of `content-server.ExtCS.nativeExtPort` to `80` and `content-server.ExtCS.sslExtPort` to `81` in the `documentum/config/configuration.yml` file. Retain the default values for all other variables.

   d. Documentum CM Server must be deployed with the externalization configuration using the preceding steps. Deploy the Documentum CM Server pod using the instructions provided in “Deploying OpenText Documentum CM using manual steps” on page 45. After the initial deployment of the pod, a load balancer service is created for `ExtCS` with an external IP address. Copy the external IP address and provide it as a value for `content-server.ExtCS.tcp_route`.

      For example:

      ```
      ExtCS:
          tcp_route:10.10.10.10
      ```

   e. Run the Helm upgrade command.

Google Cloud Platform documentation contains details information.

Chapter 3

Deploying OpenText Documentum CM on cloud platforms

This documentation provides the information to deploy OpenText Documentum CM on cloud platforms.

3.1 Deploying OpenText Documentum CM using manual steps

This section provides the information to deploy OpenText Documentum CM on cloud platforms using the manual steps.

! **Important**
All variables that are not listed in this section are optional variables which includes the non-mandatory and feature variables.

Go to “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94 **and provide the values for the variables only if required for your environment.**

To deploy OpenText Documentum CM using manual steps:

1. Perform all the relevant tasks as described in “Prerequisites for deploying and configuring OpenText Documentum CM on cloud platforms“ on page 7.

2. To enable or disable the components in the `documentum/documentum-components.yaml` file, do the following:

   a. Set the value of `enabled` to `true` for the components you want to deploy. See “Links to mandatory variables tables” on page 46.

   b. Set the value of `enabled` to `false` for the components that you do not want to deploy.

      When you set the value of `enabled` to `false` for a component, its associated `initContainers` are excluded and not deployed.

      ! **Important**
      To deploy a specific component, for example, `d2classic`, enable or disable it using the `documentum-components.yaml` file. Do not deploy the component separately using the individual Helm charts. As all the components depend on global variables defined at the root level, deploying them independently results in deployment failure.

3. Provide the appropriate values for the relevant global variables, if required for your environment, to pass them to your templates, as described in "Global variables" on page 49.

   To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

4. Provide the appropriate values for all the mandatory variables to pass them to your templates for the enabled components.

   To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

   The following table provides quick access to the list of all the mandatory variables for the enabled components:

Table 3-1: Links to mandatory variables tables

Product or component name	Links to mandatory variables tables
OpenText Directory Services	"Mandatory variables for OTDS" on page 54
OpenText Documentum CM Server	"Mandatory variables for Documentum CM Server" on page 57
OpenText Documentum CM client	"Mandatory variables for client" on page 60
Admin Console	"Mandatory variables for Admin Console" on page 61
OpenText AppWorks Gateway	"Mandatory variables for OpenText AppWorks Gateway" on page 61
OpenText Intelligent Viewing	"Mandatory variables for OpenText Intelligent Viewing" on page 63
OpenText Documentum CM Foundation CMIS API	"Mandatory variables for Foundation CMIS API" on page 65
OpenText Documentum CM Foundation SOAP API	"Mandatory variables for Foundation SOAP API" on page 66
OpenText Documentum CM Foundation REST API	"Mandatory variables for Foundation REST API" on page 67
OpenText Documentum CM Administrator	"Mandatory variables for Documentum Administrator" on page 67
OpenText Documentum CM Records Client	"Mandatory variables for Records Client" on page 68
OpenText Documentum CM Workflow Designer	"Mandatory variables for Workflow Designer" on page 69

Product or component name	Links to mandatory variables tables
OpenText Documentum CM Advanced Workflow	• "Mandatory variables for Advanced Workflow Process Integrator (BPS)" on page 70 • "Mandatory variables for Advanced Workflow xCP Deployment Agent (xDA)" on page 71
Documentum xPlore	"Mandatory variables for Documentum xPlore" on page 72
OpenText Documentum CM Messaging Service	"Mandatory variables for Messaging Service" on page 74
OpenText Documentum CM Reports	"Mandatory variables for Reports" on page 75
OpenText Documentum CM for Microsoft 365	"Mandatory variables for OpenText Documentum CM for Microsoft 365" on page 77
OpenText Documentum CM Online Editing Service	"Mandatory variables for OpenText Documentum CM Online Editing Service (OES Connector)" on page 78
OpenText Content Connect	"Mandatory variables for Content Connect" on page 80
OpenText Documentum Connector for Core Share	"Mandatory variables for Documentum Connector for Core Share" on page 83
OpenText Documentum Archive Services for SAP Solutions	"Mandatory variables for Documentum Archive Services for SAP Solutions" on page 86
OpenText Documentum Archive Services ILM for SAP Solutions	"Mandatory variables for Documentum Archive Services for SAP Solutions ILM" on page 87
OpenText Documentum Content Services for SAP Solutions	"Mandatory variables for Documentum Content Services for SAP Solutions" on page 87
Independent Java Method Server	"Mandatory variables for Independent Java Method Server" on page 88
OpenText Content Aviator	"Mandatory variables for OpenText Content Aviator" on page 88

5. Run the following command to deploy OpenText Documentum CM:

```
helm install <release name> /opt/temp/documentum --values <location where Helm
charts are extracted>/config/configuration.yml --values <location where Helm charts
are extracted>/config/constants.yaml --values <location where Helm charts are
extracted>/config/<passwords or passwords_vault or passwords_k8api>.yaml --values
<location where Helm charts are extracted>/platforms/<cloud platform>.yaml --values
<location where Helm charts are extracted>/dockerimages-values.yaml --values
<location where Helm charts are extracted>/documentum-resources-values-
```

```
<config>.yaml --values <location where Helm charts are extracted>/documentum-
components.yaml --namespace <name of namespace>
```

where, value of `<config>` can be `extra-large`, `large`, `medium`, `medium-large`, `small`, or `small-medium`.

The available resource YAML files contain pod sizing values, including CPU and memory.

Notes

- If you want to use a vault type, provide the appropriate value for `global.vaultType` in the `documentum/values.yaml` file, and then use `passwords_vault.yaml` for HashiCorp Vault or `passwords_k8api.yaml` for Kubernetes native secrets in the command instead of `passwords.yaml`.

  For more information about deploying OpenText Documentum CM with supported vault types, see“Vault” on page 155.

  Example 3-1: Helm install command to use HashiCorp Vault

  ```
  helm install dctmdeployment /opt/temp/documentum --values /opt/temp/documentum/
  config/configuration.yml --values /opt/temp/documentum/config/constants.yaml --
  values /opt/temp/documentum/config/passwords_vault.yaml --values /opt/temp/
  documentum/platforms/azure.yaml --values /opt/temp/documentum/dockerimages-
  values.yaml --values /opt/temp/documentum/documentum-resources-values-test-
  small.yaml --values /opt/temp/documentum/documentum-components.yaml --
  namespace onedctmns
  ```

  Example 3-2: Helm install command to use Kubernetes native secrets

  ```
  helm install dctmdeployment /opt/temp/documentum --values /opt/temp/documentum/
  config/configuration.yml --values /opt/temp/documentum/config/constants.yaml --
  values /opt/temp/documentum/config/passwords_k8api.yaml --values /opt/temp/
  documentum/platforms/azure.yaml --values /opt/temp/documentum/dockerimages-
  values.yaml --values /opt/temp/documentum/documentum-resources-values-test-
  small.yaml --values /opt/temp/documentum/documentum-components.yaml --
  namespace onedctmns
  ```

- If you want to use OpenText Content Aviator, run the following command:

  ```
  helm upgrade <release name> /opt/temp/documentum --values <location where Helm
  charts are extracted>/config/configuration.yml --values <location where Helm
  charts are extracted>/config/constants.yaml --values <location where Helm
  charts are extracted>/config/<passwords or passwords_vault or
  passwords_k8api>.yaml --values <location where Helm charts are extracted>/
  platforms/<cloud platform>.yaml --values <location where Helm charts are
  extracted>/dockerimages-values.yaml --values <location where Helm charts are
  extracted>/documentum-resources-values-test-small.yaml --values <location
  where Helm charts are extracted>/documentum-components.yaml --values <location
  where Helm charts are extracted>/addons/aviator/aviator-config.yaml --
  namespace <name ofnamespace>
  ```

6. Run the following command to verify the OpenText Documentum CM deployment status:

   ```
   helm status <release name>
   ```

7. Run the following command to verify the pod deployment status:

   ```
   kubectl describe pods <name of the pod>
   ```

8. Optional Run the following command to obtain the external IP address of the load balancer service of the Ingress Controller:

   ```
   kubectl get svc | grep <name of the Ingress Controller>
   ```

9. Optional Access the OpenText Documentum CM components using the external IP address of the Ingress Controller load balancer service in a browser using the following URL:

   ```
   http://<external IP address of the Ingress Controller load balancer service>/
   <OpenText Documentum CM ingress resource>
   ```

Notes

- For Documentum CM Server, ensure that Java Method Server is up and running.
- If you manually import the client configuration file into the application, ensure that you select the **Full import without actual config reset** option to avoid resetting the actual configuration.
- By default, client URLs are added as part of the deployment. You must click **Tools > Refresh Cache**, if you do not see the client URLs in client configuration.

3.1.1 Global variables

Global variables in the documentum/dockerimages-values.yaml file

- `global.repository`
- `global.fluentdImage`
- `global.appserverImageTag`
- `global.appserverImageName`
- `global.pullPolicyType`
- `global.pullSecretName`
- `global.zkStatusImage`

Global variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `docbaseOwnerPassword`

- `installOwnerPassword`
- `otdsAdminPassword`
- `databaseAdminPassword`
- `globalRegistryPassword`
- `trustStorePassword`
- `kafka_admin_user_password`
- `licenseKey`
- `msclientsecret`
- `otdsClientSecret`

Global variables in the documentum/values.yaml file

- `global.rwoStorage`
- `global.rwmstorage`
- `global.docbase`
- `global.csSecrets`
- `global.isVaultEnabled`
- `global.vaultType`
- `global.secretConfigName`
- `global.installOwnerUsername`
- `global.globalRegistryUsername`
- `global.otdsEnabled`
- `global.otdsUserName`
- `global.d2classicWebappName`
- `global.d2configWebappName`
- `global.d2restWebappName`
- `global.d2smartviewWebappName`
- `global.adminConsoleWebappName`
- `global.oesconnectorWebappName`
- `global.cryptoKeySalt`
- `global.restApiClientSessionTokenTimeout`
- `global.tomcatConnectionTimeout`
- `global.tomcatMaxHttpHeaderSize`
- `global.tomcatMaxThreads`
- `global.tomcatKeepAliveTimeout`

- `global.tomcatAcceptCount`
- `global.documentumserviceaccount`
- `global.tomcatbase_usecommonpvc`
- `global.tomcatbase_commonpvcname`
- `global.ingressUrl`
- `global.aekLocation`
- `global.connectMode`
- `global.oauthClient`
- `global.ingressProtocol`
- `global.hostShortName`
- `global.ingressDomain`
- `global.multiIngress`
- `global.publicHostShortName`
- `global.publicIngressDomain`
- `global.vpnHostShortName`
- `global.vpnIngressDomain`
- `global.webappServiceType`
- `global.proxy.enabled`
- `global.proxy.protocol`
- `global.proxy.host`
- `global.proxy.port`
- `global.proxy.noProxy`
- `global.grayLogEnable`
- `global.graylogServer`
- `global.graylogPort`
- `global.graylogTCPPort`
- `global.businessUnit`
- `global.datacenter`
- `global.environment`
- `global.persistLogs`
- `global.newrelic`
- `global.newrelicProxyHost`
- `global.newrelicProxyPort`

- `global.newrelicProxyProtocol`
- `global.sessionAllowTrustedLogin`
- `global.internalTlsEnable`
- `global.useCertificate`
- `global.ingressTLSSecret`
- `global.createCommonTruststore`
- `global.dbrServiceName`
- `global.dbr_port`
- `global.dbrConfigmapName`
- `global.dbrdataPVCName`
- `global.openshiftEnable`
- `global.openshiftTls`
- `global.externalAccessEnabled`
- `global.isCSORCLdb`
- `global.kafka_admin_user_name`
- `global.kafka_topic_name`
- `global.kafkaBrokerList`
- `global.fluentd`
- `global.fluentdTcpPort`
- `global.fluentdRestPort`
- `global.fluentdUdpPort`
- `global.eventLogLevel`
- `global.dfcRPCTracing`
- `global.markupPort`
- `global.publicationPort`
- `global.msgHost`
- `global.msgUser`
- `global.ccExtension`
- `global.dccPrefix`
- `global.cslogrotate`
- `global.jmsServiceName`
- `global.jmsPort`
- `global.secrets_Change`

- `global.removeDocumentation`
- `global.mstenantid`
- `global.msClientId`
- `global.db_hostname`
- `global.databaseusername`
- `global.dbport`
- `global.dbtype`
- `global.db_ssl`
- `global.db_ssl_mode`
- `global.dbSchema_Name`
- `global.db_service`
- `global.driverClass_Name`
- `global.amqConfigmap`
- `global.amqSecret`
- `global.fsGroup`
- `global.licenseKeyName`
- `global.logrotate_enable`
- `global.logrotate_interval`
- `global.dmsServiceName`
- `global.dmsHttpPort`
- `global.locale`
- `global.inactivityLogout`
- `global.logsStreamToConsole`

Note: The proxy variables in the `Global variables` section in the `documentum/values.yaml` file are not applicable for Documentum CM Server and Independent Java Method Server.

3.1.2 Mandatory variables for OTDS

Provide the appropriate values for all the variables. To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

OTDS variable in the documentum/documentum-components.yaml file

- `global.otds.enabled`

OTDS variables in the documentum/dockerimages-values.yaml file

- `otds.otdsws.image.source`
- `otds.otdsws.image.name`
- `otds.otdsws.image.tag`
- `otds.otdsws.image.pullPolicy`
- `otds.otdsws.image.pullSecret`
- `otds.otdsws.otdsdb.dbImgage.source`
- `otds.otdsws.otdsdb.dbImgage.name`
- `otds.otdsws.otdsdb.dbImgage.tag`
- `otds.otdsws.otdsdb.dbImgage.pullPolicy`

OTDS variables in the documentum/config/passwords.yaml file

- `otds.otdsws.crptKey`
- `otds.newrelic.NEW_RELIC_LICENSE_KEY`
- `otds.otdsws.adminPassword`
- `otds.otdsws.migration.password`
- `otds.otdsws.singleCaCert`
- `otds.otdsws.otdsdb.password`
- `otds.otdsws.otdsdb.automaticDatabaseCreation.dbAdminPassword`

OTDS variables in the documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `otds.otdsws.crptKey`
- `otds.newrelic.NEW_RELIC_LICENSE_KEY`
- `otds.otdsws.adminPassword`
- `otds.otdsws.migration.password`
- `otds.otdsws.singleCaCert`
- `otds.otdsws.otdsdb.password`
- `otds.otdsws.otdsdb.sslDBRootCert`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

OTDS variables in the documentum/config/configuration.yml file

- `otds.global.otdsUseReleaseName`
- `otds.global.namespace`
- `otds.global.database.adminDatabase`
- `otds.global.database.adminUsername`
- `otds.global.extraPodMatchLabels`
- `otds.ingress.secret`
- `otds.ingress.class`
- `otds.ingress.annotations`
- `otds.ingress.exposeIndividualEndpoints`
- `otds.ingress.paths`
- `otds.otdsws.adminEmail`
- `otds.otdsws.isBizAdmin`
- `otds.otdsws.port`
- `otds.otdsws.migration.enabled`
- `otds.otdsws.migration.usingLegacyImage`
- `otds.otdsws.migration.legacyImagePVC`
- `otds.otdsws.migration.servicePort`
- `otds.otdsws.migration.deploymentName`
- `otds.otdsws.migration.opendjUri`
- `otdsws.migration.preUpgradeJob.enabled`
- `otdsws.migration.preUpgradeJob.timeout`
- `otds.otdsws.securityContext.fsGroup`
- `otds.otdsws.securityContext.runAsGroup`
- `otds.otdsws.securityContext.runAsUser`
- `otds.otdsws.securityContext.readOnlyRootFilesystem`
- `otds.otdsws.podLabels`
- `otds.otdsws.podAnnotations`
- `otds.otdsws.statefulSet`
- `otds.otdsws.createRole`

- `otds.otdsws.serviceType`
- `otds.otdsws.serviceAnnotations`
- `otds.otdsws.carrierGradeNAT`
- `otds.otdsws.trustedProxiesRegex`
- `otds.otdsws.customSecretName`
- `otds.otdsws.targetPort`
- `otds.otdsws.protocol`
- `otds.otdsws.protocolSecret`
- `otds.otdsws.publicHostname`
- `otds.otdsws.timeZone`
- `otds.otdsws.allowDuplicateUsers`
- `otds.otdsws.allowNonIndexedSearch`
- `otds.otdsws.systemGlobalOAuthClients`
- `otds.otdsws.additionalJavaOpts`
- `otds.otdsws.enableCustomizedTruststore`
- `otds.otdsws.enableBootstrapConfig`
- `otds.otdsws.existingBootstrapConfig`
- `otds.otdsws.otdsdb.url`
- `otds.otdsws.otdsdb.useDefaultSchema`
- `otds.otdsws.otdsdb.automaticDatabaseCreation.enabled`
- `otds.otdsws.otdsdb.automaticDatabaseCreation.dbAdmin`
- `otds.otdsws.otdsdb.automaticDatabaseCreation.dbExtensions`
- `otds.otdsws.newrelic.NEW_RELIC_APP_NAME`
- `otds.otdsws.newrelic.NEW_RELIC_LOG_FILE_NAME`
- `otds.otdsws.newrelic.NEW_RELIC_LOG_LEVEL`
- `otds.otdsws.newrelic.NEW_RELIC_BROWSER_MONITORING_AUTO_INSTRUMENT`
- `otds.otdsws.newrelic.NEW_RELIC_PROXY_HOST`
- `otds.otdsws.newrelic.NEW_RELIC_PROXY_PORT`
- `otds.otdsws.pvc.enabled`
- `otds.otdsws.pvc.storage`
- `otds.otdsws.pvc.storageClassName`
- `otds.otdsws.logging.logToFiles`
- `otds.otdsws.logging.logToPVC`

- `otds.otdsws.logging.logRequests`
- `otds.otdsws.emptyDirLimits.tempDir`
- `otds.otdsws.emptyDirLimits.logsDir`
- `otds.otdsws.emptyDirLimits.workDir`
- `otds.otdsws.emptyDirLimits.initContainerDir`
- `otds.otdsws.emptyDirLimits.customTrustStoreDir`
- `otds.otdsws.vault.url`
- `otds.otdsws.vault.namespace`
- `otds.otdsws.vault.authpath`
- `otds.otdsws.vault.tokenAudience`
- `otds.otdsws.vault.proxyAddress`
- `otds.otdsws.vault.role`
- `otds.otdsws.vault.secretsPath`
- `otds.otdsws.vault.useDynamicDbSecret`
- `otds.otdsws.vault.dbBackendMountPath`
- `otds.otdsws.vault.dbBackendRole`
- `otds.otdsws.vault.agentInjector`

OTDS variables in the documentum/values.yaml file

- `otds.ingress.enabled`
- `otds.otdsws.enabled`
- `otds.otdsws.otdsdb.username`
- `otds.otdsws.serviceAccountName`
- `otds.otdsws.vault.enabled`

3.1.3 Mandatory variables for Documentum CM Server

Mandatory variables in the documentum/documentum-components.yaml file

- `global.otds.enabled`
- `global.docbroker.enabled`
- `global.content-server.enabled`
- `global.cs-secrets.enabled`
- `global.db.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `db.images.repository`

- `db.images.db.name`
- `db.images.db.tag`
- `docbroker.images.contentserver.name`
- `docbroker.images.contentserver.tag`
- `content-server.images.contentserver.name`
- `content-server.images.contentserver.tag`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `cs-secrets.docbase.password`
- `cs-secrets.contentserver.installOwner.password`
- `cs-secrets.contentserver.globalRegistry.password`
- `cs-secrets.contentserver.aek.passphrase`
- `cs-secrets.contentserver.install.appserver.admin.password`
- `cs-secrets.contentserver.install.root.password`
- `cs-secrets.database.password`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `cs-secrets.contentserver.aek.algorithm`
- `cs-secrets.database.userName`
- `content-server.docbase.id`
- `content-server.docbase.index`
- `content-server.contentserver.aek.name`
- `content-server.contentserver.localeName`
- `content-server.database.port`
- `content-server.persistentVolume.createPVC`
- `dctm-ingress.IngressPrefix`
- `dctm-ingress.ingress.class`
- `dctm-ingress.ingress.configureHost`

Note: Update the value for all instances of *`<docbase_name>`* in the `documentum/configuration.yml` file with the value provided for `global.docbase` in the `documentum/values.yaml` file.

Mandatory variables in the documentum/values.yaml file

- `global.rwoStorage`
- `global.rwmstorage`
- `global.docbase`
- `global.csSecrets`
- `global.installOwnerUsername`
- `global.globalRegistryUsername`
- `global.otdsEnabled`
- `global.otdsUserName`
- `global.aekLocation`
- `global.connectMode`
- `global.ingressProtocol`
- `global.hostShortName`
- `global.ingressDomain`
- `global.dbrServiceName`
- `global.dbr_port`
- `global.jmsServiceName`
- `global.jmsPort`
- `global.db_hostname`
- `global.databaseusername`
- `global.dbport`
- `global.dbtype`
- `global.db_service`

Note: If you want the Documentum CM Server and connection broker logs to be streamed to a standard output (`STDOUT`) or pod logs, set the value of `logsStreamToConsole` in the `documentum/values.yaml` file to `true`.

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.4 Mandatory variables for client

Mandatory variables in the documentum/documentum-components.yaml file

- `global.dctmclientinstaller.enabled`

  Note: If you set this variable to `false`, the client webapps will not be deployed.

- `global.adminconsole.enabled`
- `global.d2classic.enabled`
- `global.d2config.enabled`
- `global.d2rest.enabled`
- `global.d2smartview.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `d2config.extensionImage.name`
- `d2config.extensionImage.tag`
- `d2classic.extensionImage.name`
- `d2classic.extensionImage.tag`
- `d2smartview.extensionImage.name`
- `d2smartview.extensionImage.tag`
- `d2rest.extensionImage.name`
- `d2rest.extensionImage.tag`

Mandatory variables in the documentum/config/configuration.yml file

- `d2config.extension.createPVC`
- `d2config.customConfigurations.filename`
- `d2classic.d2report.dctm_reports_port`
- `d2classic.d2report.d2_port`
- `d2classic.d2report.d2_scheme`
- `d2classic.customConfigurations.custom`
- `d2classic.customConfigurations.hook_approach`
- `d2classic.customConfigurations.createPVC`
- `d2classic.customConfigurations.scriptPVCname`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.

- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.5 Mandatory variables for Admin Console

Mandatory variables in the documentum/documentum-components.yaml file

- `global.adminconsole.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `adminconsole.extensionImage.name`
- `adminconsole.extensionImage.tag`

Mandatory variables in the documentum/config/configuration.yml file

- `adminconsole.restApiRuntime.CookieConfiguration.enable`
- `adminconsole.serviceAccount.createserviceaccount`
- `adminconsole.customConfigurations.custom`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.6 Mandatory variables for OpenText AppWorks Gateway

Mandatory variable in the documentum/documentum-components.yaml file

- `global.appworks-gateway.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `appworks-gateway.image.repository`
- `appworks-gateway.image.awgInitContainer.repository`
- `appworks-gateway.image.appsInitContainer.repository`
- `appworks-gateway.image.appsInitContainer.tag`

Mandatory variables in the documentum/config/configuration.yml file

- `appworks-gateway.database.vendor`

- `appworks-gateway.database.server.port`
- `appworks-gateway.database.admin.user`
- `appworks-gateway.database.admin.database`
- `appworks-gateway.database.appworksdb.user`
- `appworks-gateway.database.appworksdb.database`
- `appworks-gateway.database.otds.admin.user`
- `appworks-gateway.database.otds.partition.new`
- `appworks-gateway.database.otds.partition.customPartition`
- `appworks-gateway.database.otds.resource.new`
- `appworks-gateway.database.awg.admin.newadminuser`
- `appworks-gateway.database.awg.externalurl`
- `appworks-gateway.ingress.hosts.host`
- `appworks-gateway.ingress.paths.path`
- `appworks-gateway.ingress.paths.pathType`
- `appworks-gateway.ingress.tls.enabled`
- `appworks-gateway.ingress.tls.info.secretName.name`
- `appworks-gateway.ingress.tls.info.secretName.create`
- `appworks-gateway.ingress.tls.info.secretName.key`
- `appworks-gateway.ingress.tls.info.secretName.cert`
- `appworks-gateway.ingress.tls.info.hosts`
- `appworks-gateway.secretName`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `appworks-gateway.database.admin.password`
- `appworks-gateway.database.appworksdb.password`
- `appworks-gateway.otds.admin.password`
- `appworks-gateway.awg.admin.newadminpassword`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.

- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.7 Mandatory variables for OpenText Intelligent Viewing

Mandatory variable in the documentum/documentum-components.yaml file

- `global.otiv.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `otiv.global.secretlink.image.name`
- `otiv.global.secretlink.image.tag`
- `otiv.initialization.image.name`
- `otiv.initialization.image.tag`
- `otiv.amqp.image.name`
- `otiv.amqp.image.tag`
- `otiv.viewer.image.name`
- `otiv.viewer.image.tag`
- `otiv.markup.image.name`
- `otiv.markup.image.tag`
- `otiv.config.image.name`
- `otiv.config.image.tag`
- `otiv.highlight.image.name`
- `otiv.highlight.image.tag`
- `otiv.publication.image.name`
- `otiv.publication.image.tag`
- `otiv.publisher.image.name`
- `otiv.publisher.image.tag`
- `otiv.asset.image.name`
- `otiv.asset.image.tag`

Mandatory variables in the documentum/config/configuration.yml file

- `otiv.global.adminUsername`
- `otiv.global.trustedstoreorigins`

Mandatory variables in the documentum/config/passwords.yaml

- `otiv.global.database.ivpassword`
- `otiv.global.amqp.passwords`

Mandatory variables in the documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `otiv.global.amqp.pwdKey`
- `otiv.global.otds.secretKey`
- `otiv.global.dbSecretKey`
- `otiv.global.ivDbSecretKey`
- `otiv.publication.clientSecretKey`
- `otiv.publication.monitoring.clientSecretKey`
- `otiv.highlight.clientSecretKey`
- `otiv.publisher.clientSecretKey`

Notes

- Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.
- To ensure consistent configuration across the `documentum/config/passwords.yaml`, `documentum/config/passwords_vault.yaml`, and `documentum/config/passwords_k8api.yaml` files, do the following:
  - If New Relic is disabled, keep the value of `otiv.global.newRelic.licenseKey` variable as blank.
  - If New Relic is enabled, enter a valid New Relic license key or give reference to the `licenseKey` global variable.

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.8 Mandatory variables for Foundation CMIS API

Mandatory variable in the documentum/documentum-components.yaml file

- `global.cmis.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `dctm-cmis.cmisInitContainers.name`
- `dctm-cmis.cmisInitContainers.image.name`
- `dctm-cmis.cmisInitContainers.image.tag`

Mandatory variables in the documentum/config/configuration.yml file

- `dctm-cmis.httpPort`
- `dctm-cmis.httpsPort`
- `dctm-cmis.extraConfigMountPath`
- `dctm-cmis.java.javaOptions`
- `dctm-cmis.docbroker.pvcCertSubPath`
- `dctm-cmis.mtom_status_enabled`
- `dctm-cmis.dfc.dataDir`
- `dctm-cmis.dfc.connectionMode`
- `dctm-cmis.serviceAccount.createServiceAccount`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.9 Mandatory variables for Foundation SOAP API

Mandatory variable in the documentum/documentum-components.yaml file

- `global.dfs.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `dfs.containers.initcontainer.image.name`
- `dfs.containers.initcontainer.image.tag`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `dfs.dfc.globalRegistryPassword`
- `dfs.certificate.dfcTrustStorePassword`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `dfs.single_helm.enable`
- `dfs.serviceAccount.createServiceAccount`
- `dfs.dfc.port`
- `dfs.dfc.connectionMode`
- `dfs.dfc.dataDir`
- `dfs.extensionPVC.createPVC`
- `dfs.service.ports.httpPort`
- `dfs.service.ports.sslPort`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.10 Mandatory variables for Foundation REST API

Mandatory variable in the documentum/documentum-components.yaml file

- `global.dctm-rest.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `dctm-rest.restInitContainers.name`
- `dctm-rest.restInitContainers.imageName`
- `dctm-rest.restInitContainers.imageTag`

Mandatory variables in the documentum/config/configuration.yml file

- `dctm-rest.single_helm.enable`
- `dctm-rest.serviceAccount.createserviceaccount`

Mandatory variable in the documentum/values.yaml file

- `dctm-rest.content_server.secretName`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.11 Mandatory variables for Documentum Administrator

Mandatory variable in the documentum/documentum-components.yaml file

- `global.da.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `da.images.da.name`
- `da.images.da.tag`
- `da.images.da.extensionImage.name`
- `da.images.da.extensionImage.tag`

Mandatory variable in the documentum/config/configuration.yml file

- `da.wdkAppXmlConfig.tagsnvalues`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.12 Mandatory variables for Records Client

Records Manager

Mandatory variables in the documentum/documentum-components.yaml file

- `global.otds.enabled`
- `global.records.enabled`

Mandatory variables in the documentum/config/configuration.yml file

- `records.otds.url`
- `records.otds.authentication`

Records Queue Manager

Mandatory variable in the documentum/documentum-components.yaml file

- `global.rqm.enabled`

Mandatory variable in the documentum/config/configuration.yml file

- `rqm.containers.rqm.docbrokerhostname`

Note: To enable and access the Governance and Compliance module in OpenText Documentum CM Admin Console to create and manage retention policies, you must install the Retention Policy Services DARs in the Documentum CM Server pod. Use the `initContainer` approach with the `ot-dctm-records-darinstallation` image to install the required DARs for Retention Policy Services and Records Manager.

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.

- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.13 Mandatory variables for Workflow Designer

Mandatory variable in the documentum/config/configuration.yml file

- `dctm-workflow-designer.docbaseConnection.jmsport`
- `dctm-workflow-designer.docbaseConnection.jmsprotocol`
- `dctm-workflow-designer.tomcat.httpPort`
- `dctm-workflow-designer.tomcat.httpsPort`

Mandatory variable in the documentum/config/constants.yaml file

- `dctm-workflow-designer.dbrpersistentVolume.dbrdataPVCName`
- `dctm-workflow-designer.contextPath`
- `dctm-workflow-designer.persistentVolume.pvcAccessModes`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `dctm-workflow-designer.image.pullPolicy`
- `dctm-workflow-designer.image.repository`
- `dctm-workflow-designer.image.name`
- `dctm-workflow-designer.image.tag`
- `dctm-workflow-designer.image.pullsecrets`

Mandatory variables in the documentum/config/passwords.yaml file or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `dctm-workflow-designer.docbaseConnection.truststorePassword`
- `dctm-workflow-designer.docbaseConnection.superUserPassword`
- `dctm-workflow-designer.docbaseConnection.globalRegistryPassword`
- `dctm-workflow-designer.otds.client_secret`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.

- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.14 Mandatory variables for Advanced Workflow Process Integrator (BPS)

Mandatory variable in the documentum/config/configuration.yml file

- `bps.docbaseConnection.jmsport`
- `bps.tomcat.httpPort`
- `bps.tomcat.httpsPort`

Mandatory variable in the documentum/config/constants.yaml file

- `bps.dbrpersistentVolume.dbrdataPVCName`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `bps.image.repository`
- `bps.image.name`
- `bps.image.tag`
- `bps.image.pullPolicy`
- `bps.image.pullSecrets`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `bps.docbaseConnection.truststorePassword`
- `bps.docbaseConnection.password`
- `bps.docbaseConnection.globalRegistryPassword`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/charts/bps/values.yaml file

- `bps.volumeClaimTemplates.storageClassReadWriteOnce`
- `bps.docbaseConnection.docbroker`
- `bps.docbaseConnection.port`
- `bps.docbaseConnection.jmsservicename`
- `bps.docbaseConnection.docbase`
- `bps.docbaseConnection.username`

- `bps.docbaseConnection.Domain`
- `bps.docbaseConnection.globalRegistryRepository`
- `bps.docbaseConnection.globalRegistryUsername`
- `bps.dbrpersistentVolume.dbrdataPVCName`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.15 Mandatory variables for Advanced Workflow xCP Deployment Agent (xDA)

Mandatory variables in the documentum/config/configuration.yml file

- `xda.adminConsoleServicePort`
- `xda.xdacli.username`
- `xda.xdacli.environmentMode`

Mandatory variable in the documentum/config/constants.yaml file

- `xda.dbrpersistentVolume.dbrdataPVCName`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `xda.image.name`
- `xda.image.tag`

Mandatory variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `xda.xdacli.password`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documemtum/charts/xda/values.yaml file

- `xda.docbaseConnection.docbroker`
- `xda.dbrpersistentVolume.dbrdataPVCName`

- `xda.persistentVolume.storageClassReadWriteOnce`
- `xda.image.repository`
- `xda.image.pullPolicy`
- `xda.image.pullSecrets`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.16 Mandatory variables for Documentum xPlore

Mandatory variable in the documentum/documentum-components.yaml file

- `global.xPlore-OneD.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `xPlore-OneD.image.indexserver.name`
- `xPlore-OneD.image.indexserver.tag`
- `xPlore-OneD.image.indexagent.name`
- `xPlore-OneD.image.indexagent.tag`
- `xPlore-OneD.image.cps.name`
- `xPlore-OneD.image.cps.tag`
- `xPlore-OneD.dcis-text-extractor.fetcher.image.name`
- `xPlore-OneD.dcis-text-extractor.fetcher.image.tag`
- `xPlore-OneD.dcis-text-extractor.parser.image.name`
- `xPlore-OneD.dcis-text-extractor.parser.image.tag`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `xPlore-OneD.cps.extraEnv`
- `xPlore-OneD.indexserver.extraEnv`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `xPlore-OneD.service.indexserver.name`
- `xPlore-OneD.service.indexserver.port`
- `xPlore-OneD.service.indexserver.targetport`
- `xPlore-OneD.service.indexagent.name`
- `xPlore-OneD.service.indexagent.port`
- `xPlore-OneD.service.indexagent.targetport`
- `xPlore-OneD.service.cps.name`
- `xPlore-OneD.service.cps.port`
- `xPlore-OneD.service.cps.targetport`
- `xPlore-OneD.persistentVolume.isExist`
- `xPlore-OneD.persistentVolume.claimName`
- `xPlore-OneD.persistentVolume.awsEFS`
- `xPlore-OneD.persistentVolume.awsEFSCSIDriver`
- `xPlore-OneD.persistentVolume.awsEFSCSIHandle`
- `xPlore-OneD.persistentVolume.awsEFSCSIPvName`
- `xPlore-OneD.serviceAccount.createServiceAccount`
- `xPlore-OneD.indexserver.extraEnv.NEWRELIC_HOME`
- `xPlore-OneD.indexagent.docbaseUser`
- `xPlore-OneD.indexagent.extraEnv.NEWRELIC_HOME`
- `xPlore-OneD.cps.extraEnv.NEWRELIC_HOME`
- `content-server.contentserver.fulltextEngineSSVCEnable`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.17 Mandatory variables for ActiveMQ

Mandatory variable in the documentum/documentum-components.yaml file

- `global.amq.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `amq.image.repository`
- `amq.image.name`
- `amq.image.tag`

Mandatory variables in the documentum/config/configuration.yml file

- `amq.serviceAccount.createserviceaccount`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.18 Mandatory variables for Messaging Service

Mandatory variable in the documentum/documentum-components.yaml file

- `global.dctm-dms.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `dctm-dms.dmsInitContainers.image.name`
- `dctm-dms.dmsInitContainers.image.tag`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `dctm-dms.jmx.password`
- `dctm-dms.certificates.bocsssl`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.19 Mandatory variables for Reports

Mandatory variables in the documentum/documentum-components.yaml file

- `global.dtrbase.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `content-server.extraInitContainers.name`
- `content-server.extraInitContainers.component`
- `content-server.extraInitContainers.image`
- `content-server.extraInitContainers.command`
- `content-server.extraInitContainers.volumeMounts.name`
- `content-server.extraInitContainers.volumeMounts.mountPath`
- `content-server.extraInitContainers.volumeMounts.subPath`
- `d2config.extraInitContainers.name`
- `d2config.extraInitContainers.component`
- `d2config.extraInitContainers.image`
- `d2config.extraInitContainers.command`
- `d2config.extraInitContainers.volumeMounts.name`
- `d2config.extraInitContainers.volumeMounts.mountPath`
- `d2classic.extraInitContainers.name`
- `d2classic.extraInitContainers.component`
- `d2classic.extraInitContainers.image`
- `d2classic.extraInitContainers.command`
- `d2classic.extraInitContainers.volumeMounts.name`
- `d2classic.extraInitContainers.volumeMounts.mountPath`
- `dtrbase.images.dtrbase.name`
- `dtrbase.images.dtrbase.tag`

Mandatory variables in the documentum/values.yaml file

- `global.ingressUrl`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `cs-secrets.clients.drServiceAccountPassword`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `dtrbase.containers.dtrbase.ingressHost`
- `dtrbase.containers.dtrbase.drCoreServerTimeout`
- `dtrbase.containers.dtrbase.drCoreUrlTimeout`
- `dtrbase.containers.dtrbase.drCoreReportServlet`
- `dtrbase.containers.dtrbase.drCoreReportCacheTimeout`
- `dtrbase.containers.logging.rootLoggerLevel`
- `dtrbase.containers.logging.consoleThesholdLevel`
- `dtrbase.containers.logging.maxLogFiles`
- `d2config.customConfigurations.filename`
- `d2classic.d2report.dctm_reports_port`
- `d2classic.d2report.d2_port`
- `d2classic.d2report.d2_scheme`
- `d2classic.d2report.dctm_reports_scheme`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.
- Optionally, perform the tasks as described in “Updating non-mandatory and feature variables in Helm charts (optional)” on page 94.

3.1.20 Mandatory variables for OpenText Documentum CM for Microsoft 365

Mandatory variable in the documentum/documentum-components.yaml file

- `global.smartviewm365.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `content-server.extraInitContainers.name`
- `content-server.extraInitContainers.component`
- `content-server.extraInitContainers.image`
- `content-server.extraInitContainers.command`
- `content-server.extraInitContainers.volumeMounts.name`
- `content-server.extraInitContainers.volumeMounts.mountPath`
- `content-server.extraInitContainers.volumeMounts.subPath`
- `smartviewm365.extensionImage.name`
- `smartviewm365.extensionImage.tag`
- `d2rest.extraInitContainers.name`
- `d2rest.extraInitContainers.component`
- `d2rest.extraInitContainers.image`
- `d2rest.extraInitContainers.command`

Mandatory variables in the documentum/config/configuration.yml file

- `d2rest.restApiRuntime.AllowCors.enable`
- `d2rest.restApiRuntime.AllowCors.restAllowedHeaders`
- `d2rest.restApiRuntime.AllowCors.restExposedHeaders`
- `d2rest.msgraphConfig.enable`
- `smartviewm365.readinessProbe.initialDelaySeconds`
- `smartviewm365.livenessProbe.initialDelaySeconds`
- `smartviewm365.startupProbe.initialDelaySeconds`
- `smartviewm365.ccsv.sharepointdomains`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to “Links to mandatory variables tables” on page 46.

- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

Note: Ensure that the Notification Service is enabled before enabling the OpenText Documentum CM for Microsoft 365.

3.1.21 Mandatory variables for OpenText Documentum CM Online Editing Service (OES Connector)

Mandatory variable in the documentum/documentum-components.yaml file

- `global.oesconnector.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `content-server.extraInitContainers.name`
- `content-server.extraInitContainers.component`
- `content-server.extraInitContainers.image`
- `content-server.extraInitContainers.command`
- `content-server.extraInitContainers.volumeMounts.name`
- `content-server.extraInitContainers.volumeMounts.mountPath`
- `content-server.extraInitContainers.volumeMounts.subPath`
- `oesconnector.extensionImage.name`
- `oesconnector.extensionImage.tag`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `oesconnector.configMap.database.password`
- `oesconnector.configMap.database.dbSecretKeyName`
- `oesconnector.configMap.database.dbunencryptedpassword`
- `oesconnector.configMap.database.db_ssl_rootcert`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `oesconnector.env.tomcatJVMArgs`
- `oesconnector.tomcat.connectionTimeoutInMilliseconds`
- `oesconnector.tomcat.maxHttpHeaderSize`

- `oesconnector.tomcat.logfilerotation`
- `oesconnector.tomcat.logfilesize`
- `oesconnector.dbSchemaInit.enabled`
- `oesconnector.dbSchemaInit.dbname`
- `oesconnector.configMap.database.url`
- `oesconnector.configMap.database.quartz.dbDelegateClass`
- `oesconnector.configMap.setupConfig.logLevel`
- `oesconnector.readinessProbe.initialDelaySeconds`
- `oesconnector.livenessProbe.initialDelaySeconds`
- `oesconnector.startupProbe.initialDelaySeconds`
- `oesconnector.newrelic.app_name`

Mandatory variable in the documentum/values.yml file

- `global.oesconnectorWebappName`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

Note: Ensure that the Notification Service is enabled before enabling the OpenText Documentum CM Online Editing Service.

3.1.22 Mandatory variables for Notification Service

This section provides the mandatory variables for Notification Service. Notification Service is shared between OpenText Documentum CM for Microsoft 365 and OpenText Documentum CM Online Editing Service.

Mandatory variable in the documentum/documentum-components.yaml file

- `global.notificationservice.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `notificationservice.image.name`
- `notificationservice.image.tag`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `notificationservice.configMap.database.password`
- `notificationservice.configMap.database.dbunencryptedpassword`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `notificationservice.enabled`
- `notificationservice.dbSchemaInit.enabled`
- `notificationservice.dbSchemaInit.dbname`
- `notificationservice.configMap.database.url`
- `notificationservice.configMap.extApiconfig.otdsclientid`
- `notificationservice.readinessProbe.initialDelaySeconds`
- `notificationservice.livenessProbe.initialDelaySeconds`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.23 Mandatory variables for Content Connect

Mandatory variable in the documentum/documentum-components.yaml file

- `global.contentconnect.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `content-server.extraInitContainers.name`
- `content-server.extraInitContainers.component`
- `content-server.extraInitContainers.image`
- `content-server.extraInitContainers.volumeMounts.name`
- `content-server.extraInitContainers.volumeMounts.mountPath`
- `content-server.extraInitContainers.volumeMounts.subPath`

- `contentconnect.images.cc.name`
- `contentconnect.images.cc.tag`
- `contentconnect.images.ccdb.name`
- `contentconnect.images.ccdb.tag`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `contentconnect.configmap.clientSecret`
- `contentconnect.secret.DB_PASSWORD`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `dctm-ingress.ccService.enabled`
- `dctm-ingress.ccadminService.enabled`
- `contentconnect.configmap.DB_DB`
- `contentconnect.configmap.DB_TABLESPACE_NAME`
- `contentconnect.configmap.authType`
- `contentconnect.configmap.clientId`
- `contentconnect.configmap.tenantId`
- `contentconnect.configmap.protocol`
- `contentconnect.ingress.configureHost`
- `contentconnect.ingress.tls.enable`
- `contentconnect.contentconnectdb.value`
- `contentconnect.ingress.enabled`
- `d2smartview.restApiRuntime.CookieConfiguration.enable`
- `d2smartview.restApiRuntime.ContentConnect.enable`
- `d2smartview.restApiRuntime.ContentConnect.restAllowedOrigins`
- `d2rest.restApiRuntime.AllowCors.enable`
- `d2rest.restApiRuntime.restAllowedHeaders`
- `d2rest.restApiRuntime.restExposedHeaders`

Notes

- If you are using Microsoft Sensitivity Label, make sure that you have enabled Microsoft Purview Information Protection in Documentum CM

Server for Content Connect processing. For more information, see "Sensitivity labels from Microsoft Purview Information Protection" on page 220.

- If you are using OpenText Documentum CM Branch Office Caching Services, ensure that the OpenText Documentum CM Branch Office Caching Services server is configured to run in the HTTPS mode and CORS enabled. To enable CORS in Branch Office Caching Services, update the following details in the `webapp\BOCS\WEB-INF\web.xml` file and update the public ingress URL.

```
 <filter>
   <filter-name>CORSFilter</filter-name>
        <filter-class>com.documentum.osgi.filter.CORSFilter</filter-class>
        <init-param>
            <param-name>CORSAllowed</param-name>
            <param-value>true</param-value>
        </init-param>
        <init-param>
            <param-name>AllowedHeaders</param-name>
            <param-value>Access-Control-Allow-Headers,documentum-csrf-token,
Origin, Accept, X-Requested-With, Content-Type, Access-Control-Request-Method,
Access-Control-Request-Headers, Authorization, DOCUMENTUM-CUSTOM-UNAUTH-SCHEME, X-
CLIENT-LOCATION, X-CLIENT-APPLICATION-NAME, Access-Control-Allow-Origin,x-no-
redirection</param-value>
        </init-param>
        <init-param>
            <param-name>AllowedDomains</param-name>
            <param-value><Public Ingress URL></param-value>
        </init-param>
        <init-param>
            <param-name>cors.allowed.origins</param-name>
            <param-value><Public Ingress URL></param-value>
        </init-param>
  </filter>
```

- Ensure that Foundation REST API, client, and Client REST API are configured to run in HTTPS mode.
- If you are not using client, then use Foundation REST API, see "Integrating Content Connect with Foundation REST API" on page 130.

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.24 Mandatory variables for Documentum Connector for Core Share

Mandatory variable in the documentum/documentum-components.yaml file

- `global.dctm-dcc.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `content-server.extraInitContainers.name`
- `content-server.extraInitContainers.component`
- `content-server.extraInitContainers.image`
- `content-server.extraInitContainers.command`
- `content-server.extraInitContainers.volumeMounts.name`
- `content-server.extraInitContainers.volumeMounts.mountPath`
- `content-server.extraInitContainers.volumeMounts.subPath`
- `dctm-dcc.database.dbSchemaInit.image.name`
- `dctm-dcc.database.dbSchemaInit.image.tag`
- `dctm-dcc.dcc.image.name`
- `dctm-dcc.dcc.image.tag`

Mandatory variables in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `dctm-dcc.database.password`
- `dctm-dcc.database.dbSecretKeyName`
- `dctm-dcc.database.dbunencryptedpassword`
- `dctm-dcc.syncnshareManual.configMap.setup.sourceRepoPassword`
- `dctm-dcc.syncnshareManual.configMap.setup.targetRepoClientSecret`
- `dctm-dcc.syncnshareManual.configMap.setup.sourceRepoVaultSecretName`
- `dctm-dcc.syncnshareManual.configMap.setup.targetRepoVaultSecretName`
- `dctm-dcc.syncnshareManual.configMap.database.dbSecretKeyName`
- `dctm-dcc.mailService.configMap.mailconfig.clientSecret`
- `dctm-dcc.mailService.configMap.mailconfig.mailpassword`
- `dctm-dcc.mailService.configMap.mailconfig.mailPasswordVaultSecretName`
- `dctm-dcc.metadata.database.dbSecretKeyName`
- `dctm-dcc.coreNotification.database.dbSecretKeyName`

- `dctm-dcc.coreNotification.coresharecrt`

Notes

- Provide the `clientSecret` if the authentication type is OAuth for mail service. Provide the encrypted client secret from the Microsoft Entra admin center after completing the app registration.
- Provide the `mailpassword` if the authentication type is basic and SMTP server requires credentials, then provide the encrypted password for the server.
- Provide the Core Share certificate if the `https` protocol is enabled.
- Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `dctm-ingress.dcc.syncagent.service.servicePort`
- `dctm-ingress.dcc.syncnshareManual.service.servicePort`
- `dctm-dcc.database.quartzdbDelegateClass`
- `dctm-dcc.serviceAccount.createserviceaccount`
- `dctm-dcc.dcc.configMap.serviceRegistry.serverHost`
- `dctm-dcc.metadata.prefix`
- `dctm-dcc.metadata.dbSchemaInit.enabled`
- `dctm-dcc.metadata.dbSchemaInit.dbname`
- `dctm-dcc.metadata.configMap.database.url`
- `dctm-dcc.syncagent.prefix`
- `dctm-dcc.syncagent.containerPort`
- `dctm-dcc.syncagent.dbSchemaInit.enabled`
- `dctm-dcc.syncagent.dbSchemaInit.dbname`
- `dctm-dcc.syncagent.configMap.database.url`
- `dctm-dcc.syncagent.configMap.mail.fromEmailId`
- `dctm-dcc.syncagent.configMap.mail.toEmailId`
- `dctm-dcc.syncagent.configMap.mail.bccEmailId`
- `dctm-dcc.syncagent.configMap.mail.ccEmailId`
- `dctm-dcc.syncagent.configMap.mail.subject`
- `dctm-dcc.syncagent.configMap.serviceRegistry.serverPort`
- `dctm-dcc.syncagent.service.servicePort`

- `dctm-dcc.syncagent.service.targetPort`
- `dctm-dcc.syncnshareManual.prefix`
- `dctm-dcc.syncnshareManual.containerPort`
- `dctm-dcc.syncnshareManual.dbSchemaInit.enabled`
- `dctm-dcc.syncnshareManual.dbSchemaInit.dbname`
- `dctm-dcc.syncnshareManual.javaConfigModify`
- `dctm-dcc.syncnshareManual.configMap.database.url`
- `dctm-dcc.syncnshareManual.configMap.setup.sourceRepoUser`
- `dctm-dcc.syncnshareManual.configMap.setup.sourceRepoUrl`
- `dctm-dcc.syncnshareManual.configMap.setup.targetRepoUrl`
- `dctm-dcc.syncnshareManual.configMap.setup.targetRepoClientId`
- `dctm-dcc.syncnshareManual.configMap.serviceRegistry.serverPort`
- `dctm-dcc.syncnshareManual.service.servicePort`
- `dctm-dcc.syncnshareManual.service.targetPort`
- `dctm-dcc.coreNotification.prefix`
- `dctm-dcc.coreNotification.dbSchemaInit.enabled`
- `dctm-dcc.coreNotification.dbSchemaInit.dbname`
- `dctm-dcc.coreNotification.configMap.database.url`
- `dctm-dcc.coreNotification.configMap.extApiconfig.dccroboticFQDN`
- `dctm-dcc.coreNotification.configMap.extApiconfig.dccmanualFQDN`
- `dctm-dcc.mailService.prefix`
- `dctm-dcc.mailService.configMap.mailconfig.authenticationType`
- `dctm-dcc.mailService.configMap.mailconfig.serviceProvider`
- `dctm-dcc.mailService.configMap.mailconfig.tenantId`
- `dctm-dcc.mailService.configMap.mailconfig.clientId`
- `dctm-dcc.mailService.configMap.mailconfig.tokenUrl`
- `dctm-dcc.mailService.configMap.mailconfig.graphApiUrl`
- `dctm-dcc.mailService.configMap.mailconfig.smtpmailhost`
- `dctm-dcc.mailService.configMap.mailconfig.smtpmailport`
- `dctm-dcc.mailService.configMap.mailconfig.usermail`
- `dctm-dcc.mailService.configMap.mailconfig.starttls`
- `vpningress.dcc.prefix`
- `vpningress.dcc.syncagent.prefix`

- `vpningress.dcc.syncagent.path`
- `d2publicingress.dcc.prefix`
- `d2publicingress.dcc.syncnshareManual.prefix`
- `d2publicingress.dcc.syncnshareManual.path`

Note: For the Client ID and Client secret required for Mail service, see step 15.

Mandatory variables in the documentum/values.yaml file

- `global.dccPrefix`
- `dctm-ingress.dcc.prefix`
- `dctm-ingress.dcc.syncagent.prefix`
- `dctm-ingress.dcc.syncnshareManual.prefix`

Note: When the `https` protocol is enabled, the following global variables in the `documentum/values.yaml` file must be set to `true`:

- `global.internalTlsEnable`
- `global.useCertificate`
- `global.createCommonTruststore`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.25 Mandatory variables for Documentum Archive Services for SAP Solutions

Mandatory variable in the documentum/documentum-components.yaml file

- `global.sapconnector-init.enabled`
- `global.dctm-assap.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `content-server.extraInitContainers.env.value`
- `dctm-assap.images.assap.name`
- `dctm-assap.images.assap.tag`

Mandatory variables in the documentum/config/configuration.yml file

- `dctm-assap.userProvidedServices.newrelic.appName`

3.1.26 Mandatory variables for Documentum Archive Services for SAP Solutions ILM

Mandatory variable in the documentum/documentum-components.yaml file

- `global.sapconnector-init.enabled`
- `global.dctm-assap-ilm.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `content-server.extraInitContainers.env.value`
- `dctm-assap-ilm.images.assapilm.name`
- `dctm-assap-ilm.images.assapilm.tag`

Mandatory variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `dctm-assap-ilm.ilmUserPassword`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/config/configuration.yml file

- `dctm-assap-ilm.userProvidedServices.newrelic.appName`

3.1.27 Mandatory variables for Documentum Content Services for SAP Solutions

Mandatory variable in the documentum/documentum-components.yaml file

- `global.sapconnector-init.enabled`
- `global.dctm-cssap.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `content-server.extraInitContainers.env.value`
- `dctm-cssap.images.cssap.name`
- `dctm-cssap.images.cssap.tag`

Mandatory variables in the documentum/config/configuration.yml file

- `dctm-cssap.userProvidedServices.newrelic.appName`

3.1.28 Mandatory variables for Independent Java Method Server

Mandatory variable in the documentum/documentum-components.yaml file

- `global.ijms.enabled`

Mandatory variables in the documentum/dockerimages-values.yaml file

- `ijms.images.ijms.name`
- `ijms.images.ijms.tag`

Mandatory variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `ijms.globalRegistryPassword`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault or `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.1.29 Mandatory variables for OpenText Content Aviator

Mandatory variable in the documentum/documentum-components.yaml file

- `global.dctm-dcis.enabled`

  Note: When you enable OpenText Content Aviator, Fetcher and Parser are deployed as part of the Documentum xPlore deployment.

Mandatory variables in the documentum/dockerimages-values.yaml file

- `dctm-dcis.image.name`
- `dctm-dcis.image.tag`

Mandatory variables in the documentum/config/passwords.yaml file

- `dctm-dcis.docbasePassword`
- `dctm-dcis.dfc.globalRegistryPassword`
- `dctm-dcis.activemq.password`
- `dctm-dcis.otds.oauth_client_secret`

Mandatory variables in the documentum/config/passwords_vault.yaml file

- `dctm-dcis.docbasePassword`
- `dctm-dcis.dfc.globalRegistryPassword`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_vault.yaml` file to use HashiCorp Vault.

Mandatory variables in the documentum/config/passwords_k8api.yaml file

- `dctm-dcis.docbasePassword`
- `dctm-dcis.dfc.globalRegistryPassword`
- `dctm-dcis.activemq.password`
- `dctm-dcis.otds.oauth_client_secret`

Note: Provide the appropriate values for all the preceding variables in the `documentum/config/passwords_k8api.yaml` to use Kubernetes native secrets.

Mandatory variables in the documentum/addons/aviator/aviator-config.yaml file

- `dctm-dcis.configurationFiles`
- `dctm-dcis.cs.useCSSecrets`
- `dctm-dcis.cs.secretName`
- `dctm-dcis.cs.acsServicePort`
- `dctm-dcis.dfc.dataDir`
- `dctm-dcis.dfc.docbroker`
- `dctm-dcis.dfc.port`
- `dctm-dcis.dfc.connectionMode`
- `dctm-dcis.activemq.broker_url`
- `dctm-dcis.activemq.user_name`
- `dctm-dcis.activemq.queue_name`
- `dctm-dcis.activemq.folder_queue_name`
- `dctm-dcis.csai.embedding_svc_url`

- `dctm-dcis.csai.chat_svc_url`
- `dctm-dcis.saasproxy.websocketEndpoint`
- `dctm-dcis.saasproxy.websocketAuth`
- `dctm-dcis.saasproxy.websocketConnections`
- `dctm-dcis.saasproxy.callbacksAuthEndpoint`
- `dctm-dcis.saasproxy.callbacksPermsEndpoint`
- `dctm-dcis.saasproxy.callbacksDctmAuthEndpoint`
- `dctm-dcis.otds.enabled`
- `dctm-dcis.otds.oauth_client_id`
- `dctm-dcis.otds.oauth_token_url`
- `dctm-dcis.actuatorHttpPort`
- `dctm-dcis.actuatorHttpsPort`

Next actions

- After providing the values for the mandatory variables, perform the steps from step 5 to step 9 to deploy OpenText Documentum CM.
- For quick navigation to update mandatory variables for other products and components, go to "Links to mandatory variables tables" on page 46.
- Optionally, perform the tasks as described in "Updating non-mandatory and feature variables in Helm charts (optional)" on page 94.

3.2 Deploying OpenText Documentum CM using Documentum™ Cloud Assist

This section provides the information to prepare and generate updated Helm chart using `Documentum™ Cloud Assist` and to deploy OpenText Documentum CM at the command prompt.

To prepare and generate updated Helm chart using Documentum™ Cloud Assist and to deploy OpenText Documentum CM at command prompt:

1. Download the `Documentum CM - Utilities.zip` and `Documentum CM - Utilities.tar.gz` files from My Support and extract the contents to a temporary location.
2. From the temporary location, extract the contents of the `Documentum_Cloud_Accelerator_windows-<version>.zip` and `Documentum_Cloud_Accelerator_linux-<version>.tar.gz` files.
3. Extract the contents of the `Documentum-Cloud-Assist-<version>.zip` and `Documentum-Cloud-Assist-<version>.tar.gz` files.

4. Run `dctm-cloud-assist-<version>.exe` on Windows or `dctm-cloud-assist-<version>.AppImage` on Linux.

5. On the home page in the **Deploy** tile, click **Deploy**.

6. On the **Before You Start** dialog box, complete the all the relevant prerequisite tasks such as downloading the container images, Helm charts, and so on.

7. Click **Upload Helm chart** to browse and select the `documentum` folder from the extracted Helm charts location and click **Continue**.

   ! Important
   New deployment is supported only for 25.2 and 25.4.

8. On the **Component Selection** page, do the following:

   a. In the **Cloud Platform** list, click the required cloud platform.
   b. Optional To use vault, turn on the **Secure Secrets** switch.
   c. (Only for 25.2) To configure HashiCorp Vault, turn on the **Secure Secrets** switch.
   d. (Only for 25.4) If you turn on the **Secure Secrets** switch, select a secret type.

      If you click **Kubernetes Native Secrets**, ensure that the name provided for **Secret Name** is same as you mentioned while creating a secret using the `documentum/config/vault_secret.yaml` file.
   e. By default, all the components available in the `documentum/documentum-components.yaml` file along with the following mandatory components appears:

      - **Documentum CM Server**
      - **Connection Broker**
      - **OTDS**

      Select or clear the check boxes for the relevant components based on your license and according to your requirement.

      ! Important
      You can select only either **DCM for Engineering** or **DCM for Life Sciences** along with other add-on components.

      If you select the add-on components, the configuration information related to the add-on components YAML file takes precedence instead of the information in the `documentum/values.yaml` and `documentum/config/configuration.yml` files.
   f. Click **Next**.

9. On the **Container Images** page, by default, all values from the uploaded Helm chart related to container images, image tags, repository path, and so on are pre-populated for all the variables in the following category tabs:

- **Global Variables**: Lists the global variables related to containers such as repository path, pull policy type, and so on.
- **Public Components**: Lists the container image information for the third-party components.
- **OpenText Components**: Lists the container image information such as image name and image version for the components that you enabled in the **Component Selection** page. This page also lists the subcomponents and init containers information.

Review the existing values for all the variables across all categories. You can retain the existing values or modify them or provide new values for the required variables.

To provide or modify the value for variables, move your mouse pointer over the field, click, and then provide a new value or modify an existing value. For example, you can provide or modify the value for **Repository Path**, **Image Name**, **Image Version** and so on across the category tabs.

Notes

- Modifying the component name and its type is not supported.
- Click **Previous** to go the previous page.
- Click **Cancel** to go to the home page to restart the deployment process.

10. Click **Next**.

    - If you have provided the appropriate values for all the mandatory and other variables for your requirement, the **Deployment Configuration** page is displayed.
    - If you have not provided the appropriate values for the mandatory variables, an error message is displayed.

      Click **Close** to see the error information icon displayed against the category name.

      Click the name of the category. All the mandatory variables that need appropriate values are highlighted. Provide the appropriate values and then click **Next**.

11. On the **Deployment Configuration** page, you can choose to use the default values for the variables according to your requirement for the enabled components or modify them.

    To modify the value, do one of the following:

    - Move your mouse pointer over the field, click, and modify the value.
    - Search for a variable in the **Search** box or use the search filters to modify the value.

- Use the find and replace option. Type the value of the variable in the **Search** box, click the toggle replace icon, provide the new value, and click **Replace All**.

By default in 25.4, the **Proxy** switch is enabled when you upload a Helm chart for the first time or later too. You can choose to turn on or turn off the switch according to your requirement.

12. Click **Next**.

    - If you provide the appropriate values for all the required variables, the **Chart Generation** page is displayed.
    - If you have not provided the appropriate values for the mandatory variables, an error message appears.

      Click **Close** to see the error information icon displayed against the component name.

      Click the name of the component. All the mandatory variables that need appropriate values are highlighted. Provide the appropriate values and then click **Next**.

13. On the **Chart Generation** page, click the name of the YAML file to review the updated values.

    To modify the values, click the edit icon of the YAML files to open the **Editor** page, modify the values directly or use the find and replace option, and then click **Save**.

    Note: Click **Previous** to go to the previous pages and to modify components and values.

    After reviewing the updated values, click **Generate Chart** and confirm when prompted to generate the updated Helm chart.

    Notes

    - Click **Exit Deployment** to go to the home page to restart the deployment process.
    - Click **Close** to return to the **Chart Generation** page.

14. On the **Chart Generated** dialog box, click **Continue**.

    If you did not opt for **Secure Secrets** in the **Component Selection** page, proceed to step 17.

15. (Only for 25.4) In the **Component Selection** page, if you opted for **Kubernetes Native Secrets** as the secret type, perform step 1 and step 3 in “To use Kubernetes native secrets in OpenText Documentum CM deployment:“ on page 156.

16. Click **Next**.

17. To deploy OpenText Documentum CM at the command prompt, do the following:

    a. Provide the appropriate values for the **Release Name**, **Namespace**, and **Select Resource YAML file** variable fields.

    b. Click **Generate Command** to generate the `helm install` command.

    c. Click **Copy** to copy the generated command.

    d. Open the command prompt and go to the folder containing the updated Helm chart.

    e. Run the copied Helm install command to complete the deployment process.

18. Click **Done** and then click **Exit** to exit from Documentum™ Cloud Assist.

    Note: Click **Cancel** only if you want to cancel all the changes and go to the home page to restart the deployment process.

3.3 Updating non-mandatory and feature variables in Helm charts (optional)

OpenText recommends that you update the non-mandatory and feature variables only if required for your environment. The information in this section is optional.

(Optional) To update non-mandatory variables and feature variables:

1. Provide the appropriate values for the relevant non-mandatory variables, if required for your environment. To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

2. Provide the appropriate values for the relevant feature variables, if required for your environment. To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

   For more information about the list of supported features and its variables, see “Using features in cloud platforms“ on page 155.

3. Run the following command to deploy OpenText Documentum CM:

```
helm upgrade <release name> /opt/temp/documentum --values <location where Helm
charts are extracted>/config/configuration.yml --values <location where Helm charts
are extracted>/config/constants.yaml --values <location where Helm charts are
extracted>/config/<passwords or passwords_vault or passwords_k8api>.yaml --values
<location where Helm charts are extracted>/platforms/<cloud platform>.yaml --values
<location where Helm charts are extracted>/dockerimages-values.yaml --values
<location where Helm charts are extracted>/documentum-resources-values-test-
small.yaml --values <location where Helm charts are extracted>/documentum-
components.yaml --namespace <name of namespace>
```

   where `<config>` can be `extra-large`, `large`, `medium`, `medium-large`, `small`, `small-medium`, or `test-small` resource value YAML files.

   The available resource YAML files contain pod sizing values, including CPU and memory.

Notes

- If you used `documentum-resources-values-medium.yaml` in previous deployment (for example 25.2), then you must use the same `documentum-resources-values-medium.yaml` in the `helm upgrade` command to upgrade to 25.4. Make sure that the values of the resources such as CPU, memory, and so on for all pods are equal to or greater than the values used for the upgrade process.
- If a vault type is enabled, use `passwords_vault.yaml` for HashiCorp Vault or `passwords_k8api.yaml` for Kubernetes native secrets in the command instead of `passwords.yaml`.

4. Run the following command to verify the OpenText Documentum CM deployment status:

```
helm status <release name>
```

5. Run the following command to verify the pod deployment status:

```
kubectl describe pods <name of the pod>
```

3.4 Post-deployment tasks

This section provides the information about the post-deployment tasks.

3.4.1 Licensing OpenText Documentum CM

3.4.1.1 Procuring license file from OpenText

Procure the license file from OpenText as described in OpenText Documentum Content Management License Management (https://support.opentext.com/csm?id=kb_article_view&sysparm_article=KB0834991).

3.4.1.2 Configuring OTDS and license

This section provides the information about configuring OTDS and license.

- For information about configuring OTDS using an automated method and configuring license using a manual method, see “To configure OTDS using an automated method and to configure license using a manual method:“ on page 96.
- For information about configuring OTDS and license using a manual method, see “To configure OTDS and license using a manual method:“ on page 100.

To use these instructions, ensure that you have updated the `documentum/charts/otds/charts/otdsws/config/config.yml` file to configure OTDS. If you have not updated the `config.yml` file, see “To configure OTDS and license using a manual method:“ on page 100.

To configure OTDS using an automated method and to configure license using a manual method:

1. Sign in to the OTDS Admin website using the following information:

   - URL: URL to access the OTDS Admin website.

     `<Ingress URL>/otds-admin`

   - User name: Value for the OTDS Admin user name.

     For example: `otadmin@otds.admin`

   - Password: Value provided for `otdsAdminPassword` in the `documentum/config/<passwords or passwords_vault or passwords_k8api>.yaml` file.

2. Synchronize the resources to the Documentum CM repository using the following steps:

   a. Click **Resources**.

   b. Click **<*your resource name*> > Actions > Consolidate**.

      For example: Your resource name can be `otdctmresource`.

3. Import the license file in OTDS using the following steps:

   a. On the **License Keys** page, click **Add**.

   b. On the **General** tab, provide values for the following fields:

      - **License Key Name**: Unique name.

        For example: `otdctmlicense`

      - **Resource ID**: License linked to the resource.

   c. On the **License Key** tab, click **Get License File**, browse and select the license file.

   d. Click **Save**.

4. Create a `businessadmin` user in the `otds.admin` partition using the following steps:

   a. Click **Partitions**.

   b. Click **otds.admin > Actions > View Members.**.

   c. Click **Add > New User**.

   d. On the **General** page, in the **User Name** box, type a name for this user as `businessadmin`.

   e. On the **Account** page, in the **Password Options** area, do the following:

      i. Click **Do not require password change on reset** from the list.

      ii. Clear the **User cannot change password** check box, if selected.

      iii. Select the **Password never expires** check box and click **Save**.

Additional information and tasks

Documentum CM client

Create OTDS users with the user name as `d2_mail_manager` and `d2_wf_notification_user` and keep the passwords as blank in a new non-synchronized partition which is not associated to any resources in OTDS.

By default, OTDS is enabled for client configuration. If you disable OTDS, add another user with the user name as `install_owner_user` to the same partition with the password as blank.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Select the **User cannot change password** check box.
- Select the **Password never expires** check box.

This is applicable only for the system account user partition.

Workflow Designer

If you want to use Workflow Designer with the skip SSO feature, only a user with the `install_owner` privilege can access without a license.

Advanced Workflow

Advanced Workflow is available with advanced license or as an add-on. As a user, you can build processes using Advanced Workflow but to install these processes, the xDA Documentum CM repository endpoint user must have advanced Documentum CM or an add-on license.

To start a workflow at runtime from any Documentum CM client components, you must have advanced Documentum CM or an add-on license and the required transaction capability. The transaction counter increments at runtime when a user creates a new instance of workflow irrespective of the state of the workflow.

Reports

Create an OTDS user with the user name as `dctmreports` and keep the password as blank. Ensure that the OTDS user name is same as the name provided for `drServiceAccountUser` in the `documentum/config/<passwords or passwords_vault or passwords_k8api>.yaml` file.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Select the **User cannot change password** check box.
- Select the **Password never expires** check box.

OpenText Documentum CM for Microsoft 365, OpenText Documentum CM Online Editing Service, and Notification Service

Create an OTDS user with the user name as `M365_SERVICE` with the password as `Xecmserviceuser@123`. The password is case-sensitive.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Clear the **User cannot change password** check box.
- Select the **Password never expires** check box.

Documentum Archive Services for SAP Solutions

Create an OTDS user with the user name as `installownerusername` with the password as `installownerpassword`. This is a service account.

The system automatically changes the password after the service is started.

5. Add the `businessadmin` user to the `otdsbusinessadmins` group in OTDS using the following steps:

   a. Click **Users & Groups**, and select the **Groups** tab.
   b. In the **Search** box, type `otdsbusinessadmins` to find the `otdsbusinessadmins@otds.admin` group.
   c. On the **Groups** tab, click **Actions > Edit Membership**.
   d. On the **otdsbusinessadmins@otds.admin** page, on the **Members** tab, click **Add Member**.
   e. In the **Search** box, type `businessadmin` to find the `businessadmin@otds.admin` member.
   f. Select the **businessadmin@otds.admin** check box, and click **Add Selected**.

6. Run the following command in IAPI to create the `dm_otds_license_config` object in the Documentum CM Server pod:

```
create,c,dm_otds_license_config
set,c,l,otds_url
<otds_url_including_rest>
set,c,l,license_keyname
<license_keyname>
set,c,l,business_admin_name
<business_adminname>
set,c,l,business_admin_password
<password>
save,c,l
```

   For example:

```
create,c,dm_otds_license_config
set,c,l,otds_url
http://documentumcm:8080/otdsws/rest
set,c,l,license_keyname
otdctmlicense
set,c,l,business_admin_name
businessadmin
set,c,l,business_admin_password
```

```
Password-1234567890
save,c,l
```

Notes

- Alternatively, you can create the `dm_otds_license_config` object using the Documentum Administrator graphical user interface.

  For instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC-UGD)*.

- If you modify the `dm_otds_license_config` object after the first time, you must run the `apply,c,NULL,FLUSH_OTDS_CONFIG` command in IAPI.

7. Allocate a license in OTDS to a partition using the following steps:

   a. Click **Partitions** > <*your desired partition*> > **Actions** > **Allocate to License**.

   b. On the **Allocate to License** page, click the relevant counter from the list.

      You can also allocate the license to users and groups. When you allocate a license, ensure that you allocate the relevant counter to a user or group.

   c. Click **Allocate to License**.

      Notes

      - If you modify the allocated license file after the initial deployment, you must restart OTDS.

      - You can allocate users to your license any time, but a user must exist before you can allocate the user to a license. The changes to the allocation, deallocation, and revocation take effect after 24 hours for releases prior to and including the 24.4 release.

        – For the 25.2 release, run the following command in IAPI to get the allocation, deallocation, revocation, or deletion with immediate effect:

        ```
        apply,c,NULL,FLUSH_JMS_OTDS_CACHE
        ```

        – From the 25.4 release, run the following command in IAPI to get the allocation, deallocation, revocation, or deletion with immediate effect:

        ```
        apply,c,NULL,FLUSH_OTDS_CACHE
        ```

8. On the Documentum CM Server pod, open the `otdsauth.properties` file in the `$DM_HOME\OTDSAuthLicenseHttpServerBin\config` folder and verify if information for `certificate`, `otds_rest_credential_url`, `otds_rest_ticket_url`, and `admin_username` are updated.

9. Sign in to a deployed application (for example, Documentum Administrator) with any licensed user to verify the license configuration.

   Ensure that the user authentication is successful.

To configure OTDS and license using a manual method:

1. Sign in to the OTDS Admin website using the following information:
   - URL: URL to access the OTDS Admin website.

     `<Ingress URL>/otds-admin`
   - User name: Value for the OTDS Admin user name.

     For example: `otadmin@otds.admin`
   - Password: Value provided for `otdsAdminPassword` in the `documentum/config/<passwords or passwords_vault or passwords_k8api>.yaml` file.
2. Create a non-synchronized user partition in OTDS using the following steps:
   a. Click **Partitions > Add > New Nonsynchronized User Partition**.
   b. In the **Name** box, type a name for your user partition.

      For example: `otdctmpartitions`
   c. Click **Save**.

   For information to create a synchronized user partition in OTDS, see *OpenText Directory Services - Installation and Administration Guide (OTDS250400-IWC)*.
3. Create a synchronized resource in OTDS using the following steps:
   a. Click **Resources > Add**.
   b. On the **General** page, do the following:
      i. In the **Resource name** box, type a name for the resource.

         For example: `otdctmresource`
      ii. In the **Description** box, type a description for the resource.
      iii. On the **Synchronization** tab, select the **User and group synchronization** check box and click **REST(Generic)** for **Synchronization connector**.
      iv. Click **Next**.
   c. On the **Resources > <*your resource name*> > Actions** page, select **Properties**.

      For example: Your resource name can be `otdctmresource`.
      i. On the **<*your resource name*>** page, select the **Connection Information** tab.
      ii. On the **Connection Information** page, provide the value for the following fields:
         - **Base URL**: URL endpoint for user or group provisioning REST API.

           Use the following format:

           ```
           http://<value provided for global.jmsServiceName in
           documentum/values.yaml>:<value provided for global.jmsPort
           in documentum/values.yaml>/dmotdsrest
           ```

For example: `http://dcs-pg-jms-service:9080/dmotdsrest`

- **Username**: Installation owner user name for the repository.

  Use the following format:

  `<repository name>\<installation owner user name>`

  For example: `docbase1\dmadmin`

- **Password**: Installation owner password for the repository.

d. Click **Test Connection**.

e. On the **User Attribute Mappings** tab, click **Reset to Default**.

f. For **User Attribute Mappings**, add the `client_capability` attribute with `Format` value of `2`.

   Add the `default_folder` attribute with `OTDS Attribute` value of `cn` and `Format` value of `/%s`.

g. Retain the default value for all other settings.

h. Click **Next**.

i. On the **Group Attribute Mappings** tab, click **Reset to Default**.

j. Click **Save**.

4. Click **Access Roles** and go to **Access to <***your resource name***>** (for example, `otdctmresource`).

   a. Click **Actions** and select **Include Groups**.

   b. Click **Actions** and select **View Access Role Details**.

   c. On the **User Partitions** tab, add the partition you created.

   d. Click **Save**.

5. Click **Resources > <***your resource name***> > Actions > Consolidate** to synchronize the members to your repository.

6. Ensure that the **OAuth client ID** value is the same provided for `oauthClient` in the `documentum/values.yaml` file.

   Click **OAuth clients** and create an OAuth client.

   a. For **Redirect URLs**, add the following to the **Redirect URLs** list:

      - `<Ingress URL>/D2/d2_otds.html`
      - `<Ingress URL>D2/OTDSLogoutResponse.html`
      - `<Ingress URL>/D2-Config/d2config_otds.html`
      - `<Ingress URL>/D2-Config/OTDSLogoutResponse.html`
      - `<Ingress URL>D2-Smartview/ui`
      - `<Ingress URL>oes-connector`
      - `<Ingress URL>AdminConsole`
      - `<Ingress URL>d2-rest`

   - `<Ingress URL>`

   Note: If you have disabled OTDS for client configuration, do not add the following URLs to the **Redirect URLs** list:

   - `<Ingress URL>/D2-Config/d2config_otds.html`.
   - `<Ingress URL>/D2-Config/OTDSLogoutResponse.html`.

   b. Click **Save**.

7. Click **Auth Handlers**, select **http.negotiate**, click **Action**, and select **Disable**.

8. Create roles using the following steps:

   a. Click **Partitions > <*your desired partition*> > Actions > View Members**.
   b. On the **Roles** tab, click **Add > New Role**.
   c. Add the following roles:

      - Client Capabilities: `Client_Consumer`, `Client_Contributor`, `Client_Coordinator`, and `Client_System_Administrator`.
      - User Privileges: `privilege_createtype`, `privilege_createcabinet`, `privilege_creategroup`, `privilege_sysadmin`, and `privilege_superuser`.

      Note: The role name is case-sensitive.

   d. Go to **Application Roles** to view the list of added roles.

9. Import the license file in OTDS using the following steps:

   a. On the **License Keys** page, click **Add**.
   b. On the **General** tab, provide values for the following fields:

      - **License Key Name**: Unique name.

        For example: `otdctmlicense`
      - **Resource ID**: License linked to the resource.

   c. On the **License Key** tab, click **Get License File**, browse and select the license file.
   d. Click **Save**.

10. Create a `businessadmin` user in the `otds.admin` partition using the following steps:

   a. Click **Partitions**.
   b. Click **otds.admin > Actions > View Members.**.
   c. Click **Add > New User**.
   d. On the **General** page, in the **User Name** box, type a name for this user as `businessadmin`.

e. On the **Account** page, in the **Password Options** area, do the following:

   i. Click **Do not require password change on reset** from the list.

   ii. Clear the **User cannot change password** check box, if selected.

   iii. Select the **Password never expires** check box and click **Save**.

Additional information and tasks

Documentum CM client

Create OTDS users with the user name as `d2_mail_manager` and `d2_wf_notification_user` and keep the passwords as blank in a new non-synchronized partition which is not associated to any resources in OTDS.

By default, OTDS is enabled for client configuration. If you disable OTDS, you must add another user with the user name as `install_owner_user` to the same partition with the password as blank.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Select the **User cannot change password** check box.
- Select the **Password never expires** check box.

This is applicable only for the system account user partition.

Workflow Designer

If you want to use Workflow Designer with the skip SSO feature, only a user with the `install_owner` privilege can access without a license.

Advanced Workflow

Advanced Workflow is available with advanced license or as an add-on. As a user, you can build processes using Advanced Workflow but to install these processes, the xDA Documentum CM repository endpoint user must have advanced Documentum CM or an add-on license.

To start a workflow at runtime from any Documentum CM client components, you must have advanced Documentum CM or an add-on license and the required transaction capability. The transaction counter increments at runtime when a user creates a new instance of workflow irrespective of the state of the workflow.

Reports

Create an OTDS user with the user name as `dctmreports` and keep the password as blank. Ensure that the OTDS user name is same as the name provided for `drServiceAccountUser` in the `documentum/config/<passwords or passwords_vault or passwords_k8api>.yaml` file.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Select the **User cannot change password** check box.

- Select the **Password never expires** check box.

OpenText Documentum CM for Microsoft 365, OpenText Documentum CM Online Editing Service, and Notification Service

Create an OTDS user with the user name as `M365_SERVICE` with the password as `Xecmserviceuser@123`. The password is case-sensitive.

The password is updated when the notification service pod is up and running.

In the **Password Options** area, click **Do not require password change on reset** from the list, and do the following:

- Clear the **User cannot change password** check box.
- Select the **Password never expires** check box.

Documentum Archive Services for SAP Solutions

Create an OTDS user with the user name as `installownerusername` with the password as `installownerpassword`. This is a service account.

The system automatically changes the password after the service is started.

11. Add the `businessadmin` user to the `otdsbusinessadmins` group in OTDS using the following steps:

    a. Click **Users & Groups**, and select the **Groups** tab.
    b. In the **Search** box, type `otdsbusinessadmins` to find the `otdsbusinessadmins@otds.admin` group.
    c. On the **Groups**, click **Actions > Edit Membership**.
    d. On the **otdsbusinessadmins@otds.admin** page, on the **Members** tab, click **Add Member**.
    e. In the **Search** box, type `businessadmin` to find the `businessadmin@otds.admin` member.
    f. Select the **businessadmin@otds.admin** check box, and click **Add Selected**.

12. Run the following command in IAPI to create the `dm_otds_license_config` object in the Documentum CM Server pod:

```
create,c,dm_otds_license_config
set,c,l,otds_url
<otds_url_including_rest>
set,c,l,license_keyname
<license_keyname>
set,c,l,business_admin_name
<business_adminname>
set,c,l,business_admin_password
<password>
save,c,l
```

For example:

```
create,c,dm_otds_license_config
set,c,l,otds_url
http://documentumcm:8080/otdsws/rest
```

```
set,c,l,license_keyname
otdctmlicense
set,c,l,business_admin_name
businessadmin
set,c,l,business_admin_password
Password-1234567890
save,c,l
```

Notes

- Alternatively, you can create the `dm_otds_license_config` object using the Documentum Administrator graphical user interface.

  For instructions, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC-UGD)*.

- If you modify the `dm_otds_license_config` object after the first time, you must run the `apply,c,NULL,FLUSH_OTDS_CONFIG` command in IAPI.

13. Allocate a license in OTDS to a partition using the following steps:

    a. Click **Partitions** > <*your desired partition*> > **Actions** > **Allocate to License**.

    b. On the **Allocate to License** page, click the relevant counter from the list.

    Note: You can also allocate the license to users and groups. When you allocate a license, ensure that you allocate the relevant counter to a user or group.

    c. Click **Allocate to License**.

    Notes

    - If you modify the allocated license file after the initial deployment, you must restart OTDS.
    - You can allocate users to your license any time, but a user must exist before you can allocate the user to a license. The changes to the allocation, deallocation, and revocation take effect after 24 hours for releases prior to and including the 24.4 release.

      – For the 25.2 release, run the following command in IAPI to get the allocation, deallocation, revocation, or deletion with immediate effect:

      ```
      apply,c,NULL,FLUSH_JMS_OTDS_CACHE
      ```

      – From the 25.4 release, run the following command in IAPI to get the allocation, deallocation, revocation, or deletion with immediate effect:

      ```
      apply,c,NULL,FLUSH_OTDS_CACHE
      ```

14. Import all the inline and Lightweight Directory Access Protocol (LDAP) users to OTDS:

    - On Windows:

On the Documentum CM Server pod, copy the `dfc.properties` file to the `$DOCUMENTUM\Shared` folder. At the command prompt, go to the `$DOCUMENTUM\Shared` folder, run the following command:

```
java --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/
java.io=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-
opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/
sun.rmi.transport=ALL-UNNAMED --add-exports=java.base/
sun.security.provider=ALL-UNNAMED --add-exports=java.base/
sun.security.pkcs=ALL-UNNAMED --add-exports=java.base/sun.security.x509=ALL-
UNNAMED --add-exports=java.base/sun.security.util=ALL-UNNAMED --add-
exports=java.base/sun.security.tools.keytool=ALL-UNNAMED -
cp .;dfc.jar;dfc.properties;* com.documentum.fc.tools.MigrateInlineUsersToOtds
<repository name> <installation owner user name> <installation owner password>
<non-synchronized partition name>
```

- On Linux:

  On the Documentum CM Server pod, copy the `dfc.properties` file to the `$DOCUMENTUM\dfc` folder. At the command prompt, go to the `$DOCUMENTUM\dfc` folder, run the following command:

```
java --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/
java.io=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-
opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/
sun.rmi.transport=ALL-UNNAMED --add-exports=java.base/
sun.security.provider=ALL-UNNAMED --add-exports=java.base/
sun.security.pkcs=ALL-UNNAMED --add-exports=java.base/sun.security.x509=ALL-
UNNAMED --add-exports=java.base/sun.security.util=ALL-UNNAMED --add-
exports=java.base/sun.security.tools.keytool=ALL-UNNAMED -
cp .:dfc.jar:dfc.properties:* com.documentum.fc.tools.MigrateInlineUsersToOtds
<repository name> <installation owner user name> <installation owner password>
<non-synchronized partition name>
```

15. On the Documentum CM Server pod, open the `otdsauth.properties` file in the `$DM_HOME\OTDSAuthLicenseHttpServerBin\config` folder and verify if information for `certificate`, `otds_rest_credential_url`, `otds_rest_ticket_url`, and `admin_username` are updated.

16. To verify the license configuration, sign in to any deployed application (for example, Documentum Administrator) with any licensed user.

    Ensure that user authentication is successful.

3.4.1.3 Creating new users, allocating license, and applying roles in OTDS

1. Go to **Partitions > <*partition name*> > Actions > View Members > Add > New user**.
2. Create a new user with a desired name (for example, `otdctmuser`), and set all the attributes including password.
3. Click **Save**.
4. Select the user you created and click **Actions > Allocate to License**.
5. On the **Allocate to License** dialog box, select the relevant counter and click **Allocate to License**.
6. Select the user you created and click **Actions > Edit Application Roles**.

7. Click **Assign Roles** and select a desired role for the user.

   For more information about the list of roles, see step 8.c.

   Note: Ensure `da_privilege_enabled=T` and `lss_cc_enabled=T` are added in the `<Java Method Server Home>/webapps/dmotdsrest/WEB-INF/classes/dmotds.properties` file in the Documentum CM Server pods.

8. Click **Add Selected** and then click **Close**.

3.4.1.4 Troubleshooting license configuration

The following table describes the license-related errors captured in the `$DOCUMENTUM/dba/log/otdsauth.log` file.

Error code	Description	Solution
`DM_LICENSE_E_NO_LICENSE_CONFIG`	The license configuration is not created in the repository.	Ensure that you activate the license from IAPI or Documentum Administrator.
`DM_LICENSE_E_CONFIG_MISSING_PARAMS`	One of the following mandatory parameters is not available: OTDS URL, or OTDS business administrator credentials, or License key	Ensure that you have provided valid values in IAPI or Documentum Administrator.
`DM_LICENSE_E_SERVER_NOTREACHABLE`	The OTDS URL is not reachable.	Ensure that the OTDS deployment is active and reachable.
`DM_LICENSE_E_BAD_ADMIN_CRED`	OTDS business administrator credentials are incorrect or locked.	Before you activate the license, ensure that you have provided valid values in IAPI or Documentum Administrator.
`DM_LICENSE_E_REQ_BUSINESS_ADMIN`	The OTDS business administrator user is not added to the `otdsbusinessadmins` group in OTDS.	Ensure that you add the business administrator user to the `otdsbusinessadmins` group in OTDS.
`DM_LICENSE_E_NO_LICENSE`	The license key is not configured with the license file in OTDS.	Ensure that you upload the license file in OTDS to generate the license key and provide the correct license key.
`DM_LICENSE_E_INVALID_LICENSE`	The license file is invalid.	Ensure that you have uploaded a valid license file in OTDS.

Error code	Description	Solution
`DM_LICENSE_E_USER_NOT_FOUND_OR_DUPLICATE`	The user is not found in OTDS or the user is not unique in OTDS.	Ensure that the user exists in OTDS and is unique.
`DM_LICENSE_E_USER_NO_LICENSE_ALLOCATED`	The user is not allocated with a counter.	Ensure that you allocate a counter to the user in OTDS.
`DM_LICENSE_E_USER_NO_ACCESS`	The user is not allocated to the relevant counter or all the licenses for users or transactions are consumed.	Ensure that you allocate the user to the relevant counter in OTDS.
`DM_LICENSE_E_UNEXPECTED_ERROR`	There is an unexpected error in the licensing code.	Analyze the `otdsauth.log` file to troubleshoot the issue.
`DM_LICENSE_E_SYSTEMACCT_MISUSE`	When a system account user is allocated to any other counters.	Remove the `SYSTEMACCT` or other counters.
`DM_LICENSE_E_SERVICEACCT_MISUSE`	When a service account user is allocated to any other counters.	Remove the `SERVICEACCT` or other counters.

3.4.2 ACS dependency and DNS requirements for AWS deployment

Accelerated Content Services (ACS) requires a resolvable ingress URL when a secondary Documentum CM Server pod starts, as it is responsible for accessing and transferring specific files from the primary pod. For external storage systems such as Amazon S3, ACS is also required. It works with the S3 plug-in, which then connects to the S3 store to perform all object-related operations. Hence, the ingress URL must be resolvable by creating appropriate DNS record.

When S3 is enabled:

- Documentum CM Server depends on ACS to retrieve required objects during replica pod initialization.
- ACS must be running, and its ingress URL must be resolvable through a route 53 DNS record. Documentum CM Server uses the DNS record to connect to the configured store.

When S3 is not available:

- Objects are stored locally on the file system.
- ACS is not required. Replica pods can start using the shared PVC even if ACS is not running or if no DNS record exists for the ACS ingress URL.

3.4.2.1 Creating DNS record in route 53 for AWS ingress

When you deploy ingress on AWS using a hosted domain, the `Host` and `Address` field values are displayed when you run the `kubectl get ingress` command. You must add these values as DNS records in route 53.

To create a DNS record in route 53:

1. Open **AWS Management Console** and navigate to **Route 53**.
2. Select the **Hosted Zone** for your domain.
3. Click **Create Record**.
4. Enter the **Host name** as displayed in the ingress output.
5. Set **Record type** (A for an ALB or CNAME for other endpoints).
6. In the **Value** field, enter the address displayed in the ingress. For example, Load Balancer DNS name.
7. Save the record.

After you create the DNS record, the ingress URLs can be accessed.

3.4.3 Verifying deployment of client pods

Run the `kubectl logs <pod name>` command to verify if all the client pods are deployed.

You can access the client and OTDS services through ingress using the following URLs:

- Use the following URL to access the client configuration webapp:
  ```
  {{.Values.ingressProtocol}}://{{.Values.dctm-ingress.ingress.host}}.{{.Values.dctm-
  ingress.ingress.clusterDomainName}}/D2-Config
  ```
- Use the following URL to access the client webapp:
  ```
  {{.Values.ingressProtocol}}://{{.Values.dctm-ingress.ingress.host}}.{{.Values.dctm-
  ingress.ingress.clusterDomainName}}/D2
  ```
- Use the following URL to access the Smart View webapp:
  ```
  {{.Values.ingressProtocol}}://{{.Values.dctm-ingress.ingress.host}}.{{.Values.dctm-
  ingress.ingress.clusterDomainName}}/D2-Smartview
  ```
- Use the following URL to access Client REST API:
  ```
  {{.Values.ingressProtocol}}://{{.Values.dctm-ingress.ingress.host}}.{{.Values.dctm-
  ingress.ingress.clusterDomainName}}/d2-rest
  ```
- Use the following URL to access the OTDS admin site:
  ```
  {{.Values.ingressProtocol}}://{{.Values.dctm-ingress.ingress.host}}.{{.Values.dctm-
  ingress.ingress.clusterDomainName}}/otds-admin
  ```

Note: If you are using custom client webapp names, use the following URL to access the client webapps:

```
{{.Values.ingressProtocol}}://{{.Values.dctmingress.ingress.host}}.{{.Values.dctm-
ingress.ingress.clusterDomainName}}/{{.Values.d2classicWebappName}}
{{.Values.ingressProtocol}}://{{.Values.dctmingress.ingress.host}}.{{.Values.dctm-
ingress.ingress.clusterDomainName}}/{{.Values.d2configWebappName}}
{{.Values.ingressProtocol}}://{{.Values.dctmingress.ingress.host}}.{{.Values.dctm-
ingress.ingress.clusterDomainName}}/{{.Values.d2smartviewWebappName}}
{{.Values.ingressProtocol}}://{{.Values.dctmingress.ingress.host}}.{{.Values.dctm-
ingress.ingress.clusterDomainName}}/{{.Values.d2restWebappName}}
```

3.4.4 Post-deployment tasks for Documentum Connector for Core Share

`syncagent`

1. Log in to syncagent on URL *`<dctm-ingress>`*`/syncagent/` with credentials `username: Administrator`, `password: Administrator`.
2. In **Global Settings** > **Source repository** > **Repository superuser account**, click **Edit**.
3. Type the login credentials and the Foundation REST API URL in the **Documentum URL** section.
4. Select the desired repository from the list and click **Save**.
5. Log in to Documentum Connector for Core Share.
6. In the **Security** section, select OAuth Confidential Clients.
7. In the Redirect URLs field, type **Client Description** and type *`<dctm-ingress>`*`/syncagent/oauthcallback`.
8. Copy the client secret to a Notepad file.
9. In the existing OAuth clients, find the newly created client and copy the client ID.
10. Go to *`<dctm-ingress>`*`/syncagent/`.
11. In **Global Settings** > **Target Repository**, click **Edit**.
12. Add the Core Share URL in the OpenText Core URL (Client ID and secret are the IDs and secret copied in step 8 and step 9), and click **Save**.
13. Log in to Documentum Administrator, select **Administration**, and click **User management**.
14. Click on groups, search for `dmc_sync_users`, and then add users that can use Documentum Connector for Core Share.

`syncnshare-manual`

The default name for the `sourceRepoUser` is `dccAdmin`, which must be created on OTDS after deployment. The user must be granted superuser privileges in the

Documentum CM Server, added to the `dmc_sync_users` group, and assigned the appropriate license. The user name can be customized. Any change to the user name must be reflected in the `sourceRepoUser` input and the deployment must be upgraded accordingly.

3.4.5 Improving performance in Amazon Web Services deployment

To avoid the logging overhead from the Amazon Web Services Application Load Balancer in your Amazon Web Services deployment for all OpenText Documentum CM components, turn off the following monitoring attributes in the Amazon Web Services console:

- `Access Logs`
- `Connection Logs`

3.5 Cleaning up the deployment

1. Run the following command:

```
helm del <documentumdeployment_name> --namespace <your namespace> (for helm v3)

kubectl delete pvc --all --namespace <your namespace>
```

2. Delete all PVs used in the deployment that were not deleted after clearing the PVCs.

```
# get the names of the pv's
$kubectl get pv
NAME              CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
CLAIM                      STORAGECLASS REASON   AGE
dasaris2cspv        10Gi       RWX            Retain         Bound  dasaris2/dcs-pg-
pvc           efs-sc            2d22h
dasaris2custompv    2Gi        RWX            Retain         Bound  dasaris2/
d2custom-shared-pvc  efs-sc            2d22h


# delete the pv's
$kubectl delete pv <pv-name>
persistentvolume "<pv-name>" deleted

#verify all pv's have been deleted
$kubectl get pv
No resources found.
```

3.6 Configuring an additional product- or component-specific deployment

This section provides the information for configuring an additional product- or component-specific deployment.

Perform the relevant tasks, if required for your environment, and then run the Helm upgrade command.

3.6.1 Deploying Documentum CM Server with Azure Database for PostgreSQL - Flexible server

You can deploy Documentum CM Server with Azure PaaS, `Azure Database for PostgreSQL - Flexible server`.

You must update the values of all the relevant variables in the `documentum/values.yaml` file to deploy the Documentum CM Server pod. In addition, do the following:

1. Specify the value of `cs-secrets.database.userName` in the `<database administrator user>` format in "Mandatory variables for Documentum CM Server" on page 57.
2. If you want to use the existing repository, specify the value of `cs-secrets.database.userName` in the `<repository owner name>` format in "Mandatory variables for Documentum CM Server" on page 57.

   In addition, make sure that you set the value of `content-server.docbase.existing` to `true` in the `documentum/config/configuration.yml` file.
3. Copy the Flexible server certificate from the Microsoft Azure website and provide it as a value for `database.certificate` in the `documentum/config/passwords.yaml` (if a vault type is not used) or `documentum/config/passwords_vault.yaml` (if you use HashiCorp Vault) or `documentum/config/passwords_k8api.yaml` (if you use Kubernetes native secrets) file.
4. Set the value of `global.db_ssl` to `true` in the `documentum/values.yaml` file.

   In addition, provide the appropriate value for `global.db_ssl_mode` in the `documentum/values.yaml` file.
5. Set the value of `database.paasEnv` to `false` in the `documentum/config/configuration.yml` file.

3.6.2 Configuring Documentum CM Server for Azure OAuth authentication

Documentum CM Server supports Azure OAuth 2.0 authentication.

To configure Documentum CM Server for Azure OAuth authentication:

1. In Documentum CM Server deployed on Azure, run the following IAPI commands:

```
API> retrieve,c,dm_server_config
...
3d0004d280000102
API> append,c,l,app_server_name
SET> oAuthAuthentication
...
OK
API> append,c,l,app_server_uri
SET> http://localhost:9080/oAuthAuthentication/servlet/authenticate
...
OK
API> save,c,l
```

2. Open the `oauth.properties` file located at `$DM_JMS_HOME/webapps/OTDSAuthentication/WEB-INF/classes`.

3. Retain the default values of the following variables:

   - `azure_enable=true`
   - `azure_openid_url=https://login.microsoftonline.com/common/.well-known/openid-configuration`

4. The `azure_dctm_server_app_id` variable in `oauth.properties` is the OpenText Documentum CM OAuth client ID configured in Azure. The default value is blank and you can keep the value as blank. However, OpenText recommends that you provide the configured OpenText Documentum CM OAuth client ID as the value for this parameter for enhanced security.

5. Save the `oauth.properties` file.

Note: Documentum CM Server supports authentication only using the OAuth token and does not support the authorization. In addition, the Client Credentials grant type is not supported.

3.6.3 Deploying Documentum CM Server with Oracle database

This section provides information to deploy the Documentum CM Server pod with Oracle database as a VM or Service in both the SSL and non-SSL communication modes. This section also provides information to create the default or pre-defined tablespaces in the Oracle database as a services (Amazon RDS).

To deploy Documentum CM Server with Oracle database in non-SSL communication mode:

1. Make sure that the Oracle database VM is running and the database is accessible in the non-SSL communication mode using the `1521` port.
2. Set the value of `db.enabled` to `false` in the `documentum/documentum-components.yaml` file.
3. Set the value of `isCSORCLdb` to `true` in the `documentum/values.yaml` file.
4. Set the value of `db_hostname` to *`<host name>`* or *`<service name>`* in the `documentum/values.yaml` file.
5. Change the value of `cs-secrets.docbase.password` in the `documentum/config/passwords.yaml` or `documentum/config/passwords_vault.yaml` (if you use HashiCorp Vault) or `documentum/config/passwords_k8api.yaml` (if you use Kubernetes native secrets) file. Make sure that you follow the password complexity rules. For more information about the password complexity rules, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)*.

   Note: To use HashiCorp Vault, make sure that you create the secret ID in the HashiCorp Vault server in the *`<secret_name>`*`/`*`<key_name>`* format. The value for *`<secret_name>`* is `DOCBASE_PASSWORD` and the *`<key_name>`* is the repository name.

6. Set the value of `cs-secrets.database.userName` to *`<Oracle database user name>`* in the `documentum/config/configuration.yml` file.
7. Provide a value for `databaseAdminPassword` in "Global variables" on page 49 because the `cs-secrets.database.password` variable uses the value that you specify for `databaseAdminPassword`.
8. Set the value of `global.db_service` to *`<Oracle system identifier (SID)>`* in the `documentum/values.yml` file.
9. Set the value of `content-server.database.port` to `1521` in the `documentum/config/configuration.yml` file.
10. Deploy the Documentum CM Server pod.

To deploy Documentum CM Server with Oracle database in SSL communication mode using Oracle Wallet Manager:

1. Set up the Oracle database VM in the SSL communication mode using the Oracle Wallet Manager.

   Oracle documentation contains detailed information.

   Notes

   - Use the instructions in the *Oracle* documentation and configure the server. Client configuration is not required.
   - While creating the wallet file, make sure that you use `auto_login` instead of `auto_login_local`.

   For more information about Oracle database as a VM or Service (Amazon RDS), see *Amazon Web Services* documentation.

2. Create a wallet file using the following example commands:

   ```
   orapki wallet create -wallet $ORACLE_HOME/oracle_wallet -auto_login_only
   orapki wallet add -wallet $ORACLE_HOME/oracle_wallet -trusted_cert -cert
   certificate-pem-file -auto_login_only
   ```

   where `certificate-pem-file` is the certificate bundle.

   For more information about the certificate bundle that you must download as per specific Amazon Web Services regions, see *Amazon Web Services* documentation.

3. Make sure that the Oracle database VM is running and the database is accessible in the SSL communication mode using the `2484` port.

4. Create a folder named `oracle_wallet` in a temporary location.

5. Copy the `cwallet.sso` and `ewallet.p12` files from the database machine to the `oracle_wallet` folder.

6. Provide a unique repository name for each deployment in the Oracle database.

7. Set the value of `db.enabled` to `false` in the `documentum/documentum-components.yaml` file.

8. Set the value of `isCSORCLdb` to `true` in the `documentum/values.yaml` file.

9. Set the value of `db_hostname` to *`<host name>`* or *`<service name>`* in the `documentum/values.yaml` file.

10. Change the value of `cs-secrets.docbase.password` in the `documentum/config/passwords.yaml` or `documentum/config/passwords_vault.yaml` (if you use HashiCorp Vault) or `documentum/config/passwords_k8api.yaml` (if you use Kubernetes native secrets) file. Make sure that you follow the password complexity rules. For more information about the password complexity rules, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)*.

Note: To use HashiCorp Vault, make sure that you create the secret ID in the HashiCorp Vault server in the *<secret_name>*|*<key_name>* format. The value for *<secret_name>* is `DOCBASE_PASSWORD` and the *<key_name>* is the repository name.

11. Set the value of `global.db_service` to *<Oracle system identifier (SID)>* in the `documentum/values.yml` file.
12. Set the value of `content-server.database.port` to `2484` in the `documentum/config/configuration.yml` file.
13. Set the value of `global.db_ssl` to `true` in the `documentum/values.yml` file.
14. Set the value of `content-server.oracle.p12walletFileName` to `ewallet.p12` in the `documentum/config/configuration.yml` file
15. Set the value of `content-server.oracle.ssoWalletFileName` to `cwallet.sso` in the `documentum/config/configuration.yml` file
16. Deploy the Documentum CM Server pod.
17. After the deployment starts, verify the logs in the primary Documentum CM Server pod to see a message about the wallet file.

    For example:

    ```
    Thu Apr 26 07:44:55 UTC 2022 Waiting for user to manually place the wallet files
    in /opt/dctm/data/oracle_wallet
    ```

18. When you see the preceding message, copy the wallet files from your local machine to the Documentum CM Server pod, and then run the following command:

    ```
    kubectl cp oracle_wallet <name of pod>:/opt/dctm/data -c <name of container>
    ```

19. Verify the logs in the primary Documentum CM Server pod to see a message about the wallet file.

    For example:

    ```
    Wallet Files ewallet.p12 and cwallet.sso found at /opt/dctm/data/oracle_wallet/
    ```

    The preceding message confirms about the continuation of the deployment of the pod.

3.6.4 Deploying OpenText Documentum CM Online Editing Service with Oracle database

This section provides information to deploy the OES Connector pod with Oracle database.

To provide the appropriate values, open the YAML file, and read the descriptions of the variables.

To deploy OES Connector with Oracle database in SSL communication mode using Oracle Wallet Manager:

In the `documentum/config/configuration.yml` file, do the following:

1. Provide the value for `oesconnector.configMap.database.url`.
2. Provide the value for `oesconnector.configMap.database.db_p12filename`.
3. Provide the value of `oesconnector.configMap.database.db_ssofilename`.
4. Provide the value for `oesconnector.configMap.database.quartz.dbDelegateClass`.

To deploy OES Connector with Oracle database in non-SSL communication mode:

- In the `documentum/config/configuration.yml` file, provide the value for `oesconnector.configMap.database.url` and `oesconnector.configMap.database.db_servicename`.

3.6.5 Deploying OpenText Documentum CM Online Editing Service with PostgreSQL database in SSL communication mode

This section provides the information to deploy the OES Connector pod with PostgreSQL database as a VM or Service in SSL communication modes.

To provide the appropriate values, open the YAML file, and read the descriptions of the variables.

To deploy OES Connector with PostgreSQL database in SSL communication mode:

- In the `documentum/config/passwords.yaml` file, provide the value for `oesconnector.configMap.database.db_ssl_rootcert`.

3.6.6 Deploying Notification Service with Oracle database

This section provides information to deploy the Notification Service pod with Oracle database.

To provide the appropriate values, open the YAML file, and read the descriptions of the variables.

To deploy Notification Service with Oracle database in SSL communication mode using Oracle Wallet Manager:

In the `documentum/config/configuration.yml` file, do the following:

1. Provide the value for `notificationservice.configMap.database.url`.
2. Provide the value for `notificationservice.configMap.database.db_p12filename`.
3. Provide the value for `notificationservice.configMap.database.db_ssofilename`.

To deploy Notification Service with Oracle database in non-SSL communication mode:

- In the `documentum/config/configuration.yml` file, provide the value for `notificationservice.configMap.database.url` and `notificationservice.configMap.database.db_servicename`.

3.6.7 Deploying Notification Service with PostgreSQL database in SSL communication mode

This section provides the information to deploy the Notification Service pod with PostgreSQL database as a VM or Service in SSL communication modes. The following variables are mandatory fields.

To provide the appropriate values, open the YAML file, and read the descriptions of the variables.

To deploy Notification Service with PostgreSQL database in SSL communication mode:

1. In the `documentum/config/passwords.yaml` file, provide the value for `notificationservice.configMap.database.db_ssl_rootcert`.
2. In the `documentum/config/configuration.yml` file, provide the value for `notificationservice.configMap.database.url`.

3.6.8 Deploying Content Connect with Oracle database

This section provides information to deploy the Content Connect pod with Oracle database. The following variables are mandatory fields.

To provide the appropriate values, open the YAML file, and read the descriptions of the variables.

To deploy Content Connect with Oracle database in SSL communication mode using Oracle Wallet Manager:

In the `documentum/config/configuration.yml` file, do the following:

1. Provide the value for `contentconnect.configmap.DB_SERVICE_NAME`.
2. Provide the value for `contentconnect.configmap.DB_P12_WALLET_FILE_NAME`.
3. Provide the value of `contentconnect.configmap.DB_SSO_WALLET_FILE_NAME`.
4. Provide the value of `contentconnect.configmap.DB_TNS_FILE_NAME`.

3.6.9 Deploying Content Connect with PostgreSQL database in SSL communication mode

This section provides the information to deploy the Content Connect pod with PostgreSQL database as a VM or Service in SSL communication modes. The following variables are mandatory fields.

To provide the appropriate values, open the YAML file, and read the descriptions of the variables.

To deploy Content Connect with PostgreSQL database in SSL communication mode:

1. In the `documentum/config/passwords.yaml` file, provide the value for `contentconnect.configmap.DB_SSL_CA`.
2. In the `documentum/config/configuration.yml` file, provide the value for `contentconnect.configmap.DB_SERVER_NAME`.

3.6.10 Deploying Documentum Connector for Core Share with Oracle database

This section provides the information to deploy the Documentum Connector for Core Share pod with Oracle database as a VM or Service in both the SSL and non-SSL communication modes. The following variables in `documentum/config/configuration.yml` file are mandatory fields.

To deploy Documentum Connector for Core Share with Oracle database in SSL communication mode using Oracle Wallet Manager:

In the `documentum/config/configuration.yml` file, perform the following:

1. The `dctm-dcc.database.db_sid` variable is mandatory and applicable for Oracle database.
2. Set the value of `dctm-dcc.database.db_p12_wallet_file` to `ewallet.p12`. The default value is `ewallet.p12` file name and must not be changed.
3. Set the value of `dctm-dcc.database.db_sso_wallet_file` to `cwallet.sso`. The default value is `cwallet.sso` file name and must not be changed.

To deploy Documentum Connector for Core Share with Oracle database in non-SSL communication mode:

- In the `documentum/config/configuration.yml` file, the `dctm-dcc.database.db_sid` variable is mandatory and applicable for Oracle database.

3.6.11 Deploying Documentum Connector for Core Share with PostgreSQL database in SSL communication mode

This section provides the information to deploy the Documentum Connector for Core Share pod with PostgreSQL database as a VM or Service in SSL communication modes. The following variables are mandatory fields.

To deploy Documentum Connector for Core Share with PostgreSQL database in SSL communication mode

1. In the `documentum/config/configuration.yml` file, perform the following:
   a. Set the value of `dctm-dcc.database.sslrootcertRequired` as follows:
      - `True` when `sslmode` is set as `verify-ca` or `verify-full`.
      - `True` or `false` when `sslmode` is set `prefer` or `require`.

      The default value is `true`.
   b. Set the value of `dctm-dcc.database.sslcertRequired` to `true` when SSL database server requires valid client certificate. Provide the client certificate `sslcert` and key `sslkey`. The default value is `false`.

2. In the `documentum/config/passwords.yaml` file, perform the following:

    a. In the `dctm-dcc.database.sslrootcert`, provide the `root_ca` certificate when `sslmode` is enabled for the PostgreSQL database.

    b. In the `dctm-dcc.database.sslcert`, provide the client certificate when `sslmode` is enabled and client certificate is used for the PostgreSQL database.

    c. In the `dctm-dcc.database.sslkey`, provide the key when `sslmode` is enabled and client certificate is used for the PostgeSQL database.

3.6.12 Deploying Documentum Connector for Core Share with proxy enabled

This section provides the information to deploy the Documentum Connector for Core Share pod with proxy enabled. The following variables in `documentum/values.yaml` file are mandatory fields.

To deploy Documentum Connector for Core Share with proxy enabled:

In the `documentum/values.yaml` file, perform the following:

1. Set the value of `global.proxy.enabled` to `true` .

2. If the proxy in enabled, provide the protocol , proxy host name, and proxy port values to `global.proxy.protocol`, `global.proxy.host` and `global.proxy.port` respectively.

3. If the proxy is enabled, provide the required Foundation REST API, Client REST API, Smart View, OTDS internal service URLs for noproxy inputs to `global.proxy.noProxy`. The entries are separated by comma (`,`).

3.6.13 Creating PostgreSQL database instance on Google Cloud Platform

1. Create the PostgreSQL database instance on Google Cloud Platform.

    Google Cloud Platform documentation contains detailed information.

2. Do the following:

    a. Provide the details of database created in Google Cloud Platform in the `database` section in the `documentum/config/configuration.yml` and `documentum/values.yaml` files.

    b. Provide the details for `database.userName` (the value must be `postgres` only if you are using PostgreSQL database irrespective of instance or host name) in the `documentum/config/configuration.yml`.

    c. Provide the details for `cs-secrets.database.password` in the `documentum/config/passwords.yaml` file.

Notes

- To use HashiCorp Vault, provide the details for `cs-secrets.database.password` in the `documentum/config/passwords_vault.yaml` file.
- To use Kubernetes native secrets, provide the details for `cs-secrets.database.password` in the `documentum/config/passwords_k8api.yaml` file.

3. Deploy the Helm.
4. Run the following command to verify the OpenText Documentum CM deployment status:

```
helm status <release name>
```

5. Verify the pod deployment status.

3.6.14 Configuring Elastifile file system on Google Cloud Platform

Elastifile Cloud File System (ECFS) is a file system that can be scaled to a large number of users. ECFS is POSIX-compliant NFS file services.

ECFS supports all the enterprise-class storage features. In addition, it supports the following features:

- De-duplication
- Compression
- Snapshots
- Cross-region replication
- Quotas

Using ECFS, you can leverage Google Cloud Platform to shift, burst, accelerate, and/or scale file system applications and workflows without refactoring the application.

ECFS is used in two modes:

- Fully managed ECFS instances
- Self managed ECFS instance and provisioner

To integrate, use the dynamic storage provisioner to define `StorageClass` and `PVC`.

Example of the `documentum/values.yaml` file:

```
storageClass: ecfs-rwx
awsEFS: false
awsEFSCSIDriver: efs.csi.aws.com
awsEFSCSIHandle: fs-82630bfa
csdataPVCName: dcs-data-pvc
createPVC: true
```

```
existVolumePv:

volumeClaimTemplate:
vctName: documentum-vct
storageClass: ecfs-rwx
logvctStorageClass: ecfs-rwx
```

3.6.14.1 Configuring ECFS in fully managed mode

1. Make sure that you have the ECFS access permission to enable the services. *Google Cloud Platform* documentation contains detailed information.

2. Search and navigate to the **Elastfile File Service** page and click **CREATE** to create an instance. Provide all the mandatory details and click **CREATE**.

3. After the instance is created, record the IP address of the NFS server and the root path that must be configured as NFS service in **Share Name**. Use this to configure PV or PVC. Example NFS server configuration:

   ```
   nfs:
   path: /ecfs-poc-flymngdsvc/root/test-dbr-vct-sjtestdbr-1-pvc-
   db0a5969-1fcd-11ea-8805-42010a8001a0
   server: 192.168.0.1
   ```

4. Optional Download the `nfs-client-provisioner` Helm chart from the Github website and run the following command to install the chart:

   ```
   helm install stable/nfs-client-provisioner
   --set nfs.server=<x.x.x.x
   --set nfs.path=</exported/path>
   --set nfs.name=<customStorageClassName>
   ```

   For example:

   ```
   helm install stable/nfs-client-provisioner
   --set nfs.server=<192.168.0.1
   --set nfs.path=/ecfs-poc-flymngdsvc/root/
   --set nfs.name=ecfs-fully-managed-rwx
   ```

3.6.14.2 Configuring ECFS in self managed mode

1. Perform all the steps to install ECFS using Google Marketplace. In addition, configure and deploy ECFS. *Google Cloud Platform* documentation contains detailed information.

2. To avoid the networking issues, make sure that the value provided for **ZONE** and **Network Name** is same as mentioned in the GKE cluster where your application must be deployed.

3. Retain all the other default values as described in the *Google Cloud Platform* documentation except for the values related to storage.

4. Set up Elastifile storage provisioner. *Google Cloud Platform* documentation contains detailed information about:

   - Deploying Elastifile Kubernetes provisioner
   - Deploying Kubernetes provisioner in Google Cloud Platform for use with an ECFS

Notes

- If your Kubernetes provisioner requires network access to your ECFS cluster, it is recommended that Elastifile share the same VPC and subnet.
- For Management console URL, user name and password, use the values from step 1.
- The default value of `storageClass` is `elastifile` and cannot be changed.
- Create a PVC with annotation as `volume.beta.kubernetes.io/storage-class: "elastifile"` to validate the configuration. If required, modify the size of `storage` and/or the mode of `accessModes`.

3.6.15 Configuring a custom client during deployment

3.6.15.1 Creating and deploying a custom client configuration container image

To have a custom configuration during deployment, you must create a container image with the custom configuration included as a ZIP or XML file. You must download the busybox image from the respective website.

To create and deploy a custom client configuration container image:

1. Copy the following entries to a Docker file:

```
    FROM <repository IP address>:5000/busybox:1.35
    ARG CUSTOM_FILE_NAME
    RUN adduser -D -H dmadmin &&\
        mkdir -p /opt/D2-install/custom && \
        chown -R dmadmin:dmadmin /opt/D2-install/custom

    COPY --chown=dmadmin:dmadmin $CUSTOM_FILE_NAME /opt/D2-install/custom/
    CMD sh
```

2. Modify the busybox image (`<repository IP address>:5000/busybox:1.35`) to the downloaded busybox image in the local environment.

3. Run the following command to build the container image:

```
docker build -f Dockerfile -t <repository IP address>/ConfigCustom:latest --build-
arg CUSTOM_FILE_NAME=<custom config xml or zip file> --no-cache .
```

   For example:

```
docker build -f Dockerfile -t 10.192.02.01:5000/D2ConfigCustom:latest --build-arg
CUSTOM_FILE_NAME=docbase1-2020-07-16-Export-Config.xml  --no-cache .
```

4. After the container image is built, add the respective values in the `d2config` section in the `documentum/config/configuration.yml` file. When you import client configurations in a multiple repository deployment, if there are common config files to be imported across repositories, add the common config files in the `default` variable as shown in the following example and then update the file name:

   - If there are no common config files to be imported across repositories, the `default` variable is not required.

- If there are multiple config files to be imported for each repository, add the file names separated by comma.

  For example:

  ```
  d2config:
    customConfigurations:
    custom: true
    filename:
    - default:config1.zip,config2.xml
    - docbase1:config1.zip,config2.xml,config3.zip
    - docbase2:config1.zip,config2.xml,config3.zip
  ```

- For a single repository deployment, include the config files for your repository.

  For example:

  ```
  d2config:
    customConfigurations:
    custom: true
    filename:
    - docbase:config1.zip,config2.xml
    d2configImportParameters: "-full_import true -reset false"
  ```

Notes

- Config files must be imported only in the XML or ZIP formats.
- If you manually import client configurations, ensure that you do not overwrite the client URLs that are being updated.
- Before accessing client configuration, check the client configuration pods to ensure that multiple config files are successfully imported.

5. Add the following values in the `d2config` section in the `documentum/config/configuration.yml` file:

```
d2config:
  ### Custom Configuration ###
  customConfigurations:
    custom: True
    filename: <name of the client configuration file>
```

3.6.15.2 Deploying custom DAR files during client deployment

To deploy a custom DAR file during the client pod deployment, you must create a container image with a custom DAR file. You must download the busybox image from the respective website.

Note: If you want to deploy multiple custom DAR files in a sequence, create a text file with the name `darOrder.txt` and update the file with the repository name followed by a colon and then the DAR file names in the order they should be deployed in a comma-separated manner. If you have multiple repositories, add a new row and update the DAR files for each repository.

For example:

```
docbase1:dar1.dar,dar2.dar,dar3.dar
docbase2:dar4.dar,dar5.dar
docbase3:dar6.dar,dar7.dar
```

Place the `darOrder.txt` file in the same location as other custom DAR files and create the container image. Make sure to have the `darOrder.txt` file in UNIX format before creating the container image.

To deploy a custom DAR or multiple DAR files during the client deployment:

1. Copy the following entries to a Docker file:

   ```
   FROM <repository IP address>:5000/busybox:1.35
   RUN adduser -D -H dmadmin &&\
   mkdir -p /opt/D2-install/custom && \
   chown -R dmadmin:dmadmin /opt/D2-install/custom

   COPY --chown=dmadmin:dmadmin ./*.dar /opt/D2-install/custom/
   CMD sh
   ```

2. Modify the busybox image (`<repository IP address>:5000/busybox:1.35`) to the downloaded busybox image in the local environment.

3. Run the following command to build the container image:

   ```
   docker build -f Dockerfile -t <repository IP address>/D2CustomDar:latest  --no-
   cache .
   ```

   For example:

   ```
   docker build -f Dockerfile -t 10.192.02.01:5000/D2CustomDar:latest --no-cache .
   ```

4. Add the following image details to the `extraInitContainers` section of `content-server` in the `documentum/dockerimages-values.yaml` file:

   ```
   content-server:
     extraInitContainers:
      - name: d2customdarinit
        image: <custom_dar_image_name_with_tag>
        command: ['/bin/sh', '-c', 'yes | sudo cp -rf /opt/D2-install/custom/* /opt/
   dctm_docker/customscriptspvc/d2']
         volumeMounts:
         - name: dcs-data-pvc
           mountPath: /opt/dctm_docker/customscriptpvc
           subPath: initcontainercustomscripts/dcs-pg
   ```

3.6.15.3 Creating and deploying a custom DAR and client configuration with a single container image

You can deploy both the custom configuration for client and the custom DAR files in a single Docker file.

Note: If you want to deploy multiple custom DAR files in a sequence, create a text file with the name `darOrder.txt` and provide the names of the custom DAR files in a new line with .dar extension. Place the `darOrder.txt` file in the location where you have the other custom DAR files and create the container image.

To create and deploy a custom DAR and client configuration with a single container image:

1. Copy the following entries to a Docker file:

   ```
   FROM <repository IP address>:5000/busybox:1.35
   ARG CUSTOM_FILE_NAME
   ARG CUSTOM_DAR_FILE
   RUN adduser -D -H dmadmin &&\
   mkdir -p /opt/D2-install/custom && \
   chown -R dmadmin:dmadmin /opt/D2-install/custom

   COPY --chown=dmadmin:dmadmin $CUSTOM_FILE_NAME /opt/D2-install/custom/
   COPY --chown=dmadmin:dmadmin $CUSTOM_DAR_FILE /opt/D2-install/custom/
   CMD sh
   ```

2. Modify the busybox image (`<repository IP address>:5000/busybox:1.35`) to the downloaded busybox image in the local environment.

3. Run the following command to build the container image:

   ```
   docker build -f Dockerfile -t <repository IP address>/D2CustomDar:latest --build-
   arg CUSTOM_FILE_NAME=<custom config xml or zip file> --build-arg
   CUSTOM_DAR_FILE=<custom DAR file> --no-cache .
   ```

   For example:

   ```
   docker build -f Dockerfile -t 10.192.02.01:5000/D2ConfigCustom:latest --build-arg
   CUSTOM_FILE_NAME=docbase1-2020-07-16-Export-Config.xml --build-arg
   CUSTOM_DAR_FILE=1.D2-Base.dar --no-cache .
   ```

4. After the container image is built, add the respective values in the `d2config` section in the `documentum/config/configuration.yml` file. When you import client configurations in a multiple repository deployment, if there are common config files to be imported across repositories, add the common config files in the `default` variable as shown in the following example and then update the file name:

   - If there are no common config files to be imported across repositories, the `default` variable is not required.
   - If there are multiple config files to be imported for each repository, add the file names separated by comma.

     For example:

     ```
     d2config:
       customConfigurations:
       custom: true
       filename:
       - default:config1.zip,config2.xml
       - docbase1:config1.zip,config2.xml,config3.zip
       - docbase2:config1.zip,config2.xml,config3.zip
     ```

   - For a single repository deployment, include the config files for your repository.

     For example:

     ```
     d2config:
       customConfigurations:
       custom: true
       filename:
     ```

```
- docbase:config1.zip,config2.xml
d2configImportParameters:"-full_import true -reset false"
```

Notes

- Config files must be imported only in the XML or ZIP formats.
- If you manually import client configurations, ensure that you do not overwrite the client URLs that are being updated.
- Before accessing client configuration, check the client configuration pods to ensure that multiple config files are successfully imported.

5. Add the following values in the `d2config` section in the `documentum/config/configuration.yml` file:

```
d2config:
  ### Custom Configuration ###
  customConfigurations:
    custom: True
    filename: <name of client configuration file>
```

3.6.16 Set the acs_base_url to localhost and acs_supported_protocol to s3 for S3 store/Google Cloud Store

1. Make sure the value of `disableUpdateAcsUrl` is set to `true` for `content-server` in `documentum/values.yaml` or `documentum/config/configuration.yml`.

2. Open IAPI in the Documentum CM Server pod and run the following commands:

```
API>?,c,select object_name,r_object_id from dm_acs_config
#Note down the acs_base_url by dumping any one of the ACS r_object_id & make sure
you update
all the returned ACS r_object_id using following commands
API> fetch,c,<id1 returned above>
API> set,c,l,acs_base_url[0]
SET> http://localhost:9080/ACS/servlet/ACS
API> set,c,l,acs_supported_protocol
SET> s3
API> append,c,l,acs_base_url
<original ingress URL>
API> append,c,l,acs_supported_protocol
https
API> save,c,l
 …
OK
#To check if the parameter is updated, open a new iapi session and run the
following command:
API> ?,c,select object_name,r_object_id from dm_acs_config
#Dump each <r_object_id> and check the acs_base_url and acs_supported_protocol
parameter
dump,c,<r_object_id>
# The 0th index of acs_base_url and acs_supported_protocol has to be localhost and
s3 correspondingly
```

3.6.17 Configuring Content Connect

3.6.17.1 Enabling OTDS for Content Connect

1. Open `http://<dctm-ingress>/otds-admin`, and go to the **OAuth Clients** section and click the OAuth client used for client redirect URLs.
2. Click **Next** and go to the **Redirect URLs** section.
3. Click **Add** and specify the Content Connect URL.

   For example: `https://<dctm-ingress>`
4. Click **Save**.
5. Add the following sites to the **Trusted Sites** section:

   ```
   https://<dctm-ingress>
   https://*.sharepoint.com
   https://outlook.office.com
   https://*.officeapps.live.com/
   https://outlook.office365.com
   ```
6. Save the configuration.

3.6.17.2 Content Connect Admin Console configuration

Log in to the application using the following URL format:

```
https://<host>.<dctm-ingress URL>/<ccExtension>/admin
```

where `<dctm-ingress URL>` is the OpenText Documentum CM Ingress host mentioned in the `dctm-ingress` section of `documentum/values.yaml` and `<ccExtension>` is the value provided for `ccExtension` in the common variables section.

For example:

```
https://<host>.<dctm-ingress>/otdsws
```

1. Provide the default user credentials as `SystemAdmin/SystemAdmin@123`. When you first log in to the Content Connect Admin Console, it is mandatory to change the default password for the `SystemAdmin` user. In addition, change the password for the `BusinessAdmin` user. The default password for this user is `BusinessAdmin@123`.
2. Log in to the Admin Console with the new password.
3. The `Authentication type` is set to `OTDS` by default, you can modify the authentication as required and provide the OTDS Server information.

   For example: `https://`*`<host>`*`.`*`<otds-ingress>`*`/otdsws` and the OTDS Client ID.
4. Select endpoints as `Documentum-D2-REST-Server` and provide the Client REST API endpoint. If you are using Foundation REST API then use the Foundation REST API URL.

For example: `https://<host>.<ingress-url>/d2-rest`.

5. Click **Test** to test the connection.

   You can either point to Classic View or Smart View.

   - To point to Classic View, provide the client URL (for example, `https://<host>.<ingress-url>/D2`).
   - To point to Smart View, select the **Documentum Smart View** check box and provide the Smart View URL (for example, `https://<host>.<ingress-url>/D2-Smartview`).

6. Select **Documentum** in the Office online settings section. This is optional and must be enabled for Word, Excel, and PowerPoint online applications.

7. Click **Save**.

8. Select **Enable proxy server** and provide the proxy server URL according to your environment.

9. Click **Download manifest** to download the Content Connect manifest files for Outlook and Word from the Admin Console and Microsoft Office from the Content Connect Admin Console.

10. In client configuration, select **Tools** > **Options** > **Allowed actions in URL**, and select the following options to enable intelligent URL actions:

    - `D2_ACTION_CONTENT_IMPORT`
    - `D2_ACTION_DISPLAY_DIALOG`
    - `D2_ACTION_CONTENT_CHECKIN`
    - `D2_ACTION_FOLDER_CREATE`
    - `D2_ACTION_CONTENT_IMPORT_STRUCTURE`
    - `D2_ACTION_CONTENT_VIEW`

11. Use the manifest file to add the Content Connect add-in to the Microsoft Office or Outlook application. For more information, see *OpenText Content Connect - Deployment and Administration Guide (EDCCO250400-IGD)*.

3.6.17.3 Integrating Content Connect with Foundation REST API

Note: These steps are applicable only if you are not using client.

1. If Foundation REST API is not deployed, then do the following:

   a. Go to the `charts\dctm-rest\templates`, open the `configMap-rest-api-runtime-properties.yaml` file.

   b. Add the following entries:

   ```
   rest.cors.enabled=true
   ```

```
rest.security.client.token.cookie.samesite=None; Secure; Partitioned
rest.cors.allowed.origins=<dctm-ingressurl>
rest.cors.allowed.methods=GET, POST, PUT, DELETE, OPTIONS, HEAD
rest.cors.allowed.headers=DOCUMENTUM-CUSTOM-UNAUTH-SCHEME, Authorization,
Content-Type, Accept, X-CLIENT-LOCATION, X-CLIENT-APPLICATION-NAME, OT-DCTM-
PRODUCT-CODE, x-no-redirection
rest.cors.exposed.headers=Accept-Ranges,Content-Encoding,Content-Length,
Content-Range, Authorization, Content-Disposition
rest.dql.access.mode=all
```

c. Save the changes.

2. If Foundation REST API is already deployed, then do the following:

   a. Run the following command to edit the ConfigMap of `rest-api-runtime-properties`.

   ```
   kubectl edit cm rest-api-runtime-properties
   ```

   b. Add the following entries:

   ```
   rest.cors.enabled=true
   rest.cors.allowed.origins=<Content Connect URL>
   rest.cors.allowed.methods=GET, POST, PUT, DELETE, OPTIONS, HEAD
   rest.cors.allowed.headers=Access-Control-Allow-Origin, DOCUMENTUM-
   CUSTOMUNAUTH- SCHEME, Authorization, Content-Type, Accept, X-CLIENT-LOCATION,
   XCLIENT- APPLICATION-NAME, OT-DCTM-PRODUCT-CODE, x-no-redirection
   rest.cors.exposed.headers=Accept-Ranges, Content- Encoding,Content-Length,
   Content-Range,Authorization, Content- Disposition
   rest.should.parse.emails=true
   rest.security.client.token.cookie.samesite=None; Secure; Partitioned
   ```

   c. Save the changes.

   d. Restart the Foundation REST API pod.

3.6.18 Configuring xDA environment for Advanced Workflow

3.6.18.1 Enabling the xDA environment

1. Open the `documentum/config/configuration.yml` file and provide appropriate value for `xdacli.username` and `xdacli.environmentMode`.

2. Open the `documentum/config/passwords.yaml` file and provide appropriate value for `xdacli.password`.

3. Open the `documentum/charts/xda/templates/xda-config-configmap.yaml` file and provide appropriate value for `serviceName` and `port`.

4. Log in to the xDA user interface.

5. Go to the **Environments** tab and ensure that the environment status for the deployed environment is set to **Provisioned**.

3.6.18.2 Configuring Virtual Machine based deployment environment

1. Download the latest Advanced Workflow build from My Support and extract the contents into a virtual machine.
2. Create a process template in Advanced Workflow.
3. Open the **Preferences** dialog box and click **Deployment** > **Deployment Environments**.
4. In the **Deployment Environments** dialog box, click **Add**.
5. In the **Add Deployment Environment** dialog box, provide the following details:

Name	Description
`Environment Name`	Name of the deployment environment.
`Protocol`	Communication protocol.
`Host name`	xDA host name.
`Port number`	xDA port number.
`Username`	xDA user name.
`Password`	xDA password.

6. Click **Test Connection** to verify the credentials. If the connection is successful, the xDA environment that you created is listed in the **xDA Environment** list.

Notes

Before testing the connection, make sure that:

- The xDA pod is running and the ingress URL is accessible.
- The xDA environment is in the **Provisioned** state.
- The user name and password are correct.

7. Click **Finish**.
8. In the **Preferences** dialog box, click **Run Configuration** and click **Apply Now**.
9. In the **Run Configurations** dialog box, click **Add** to open the **Add Run Configuration** dialog box.
10. In the **Add Run Configuration** dialog box, type a run configuration name in the **Name** box and click **Next**.
11. On the **xPlore indexing configuration** page, click **Next**.
12. On the **Configure Parameters and Endpoints** page, click **Finish**.
13. In the **Run Configurations** dialog box, click **Apply and Close**.
14. Deploy the application using the **Deploy** option in Advanced Workflow to complete the process.

3.6.19 Configuring OpenText Documentum CM for Microsoft 365

1. Deploy Client REST API with Documentum CM Server.
2. Extract the `D2-config.zip` file. The extracted folder contains the `Teams-config.zip` file.

   Note: Use the `D2-config.zip` file which is downloaded from My Support.

3. In client configuration, import the `Teams-config.zip` file and select the following options:
   - **FOLDER** and **xecmdms_teams_Acl** for the security configuration element
   - **xecmdms_channel** and **xecmdms_teams** for the context
4. Configure the following security configuration elements:
   - For the **xecmdms_channel** context, enable the **FOLDER** security configuration element.
   - For the **xecmdms_teams** context, enable the **xecmdms_teams_Acl** security configuration element.
5. Refresh the client configuration cache.
6. Deploy and configure the OTDS service for Documentum CM Server and Client REST API.
7. Complete the instructions in “To configure OTDS and license using a manual method:“ on page 100.
8. Create the OTDS users with the Microsoft Teams user IDs mapped as part of the email configuration.
9. Ensure that the administrator creates the user in OTDS with the user name as `M365_SERVICE` and password as `Xecmserviceuser@123` (password is case-sensitive). Make sure in the **Password Options** area, from the list, you select **Do not require password change on reset**, and do the following:
   - Clear the **User cannot change password** check box.
   - Select the **Password never expires** check box.

   For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS250400-IWC)*.

   Note: Ignore this if you have already performed this step as part of Licensing OpenText Documentum CM. Make sure to allocate the user `M365_SERVICE` to the appropriate license in OTDS.

10. Configure OAuth Clients on OTDS for OpenText Documentum CM for Microsoft 365.

a. Open `http://<dctm-ingress>/otds-admin`, and go to the OAuth Clients section and click the OAuth client used for client redirect URLs.
b. Click **Next** and go to the Redirect URLs section.
c. Click **Add** and specify the Ingress URL.
d. Click **Save**.
e. Add the following sites to the Trusted Sites section:

   - `https://<dctm-ingress-url>`
   - `https://<dctm-ingress-url>/d2-rest`
   - `https://teams.microsoft.com`

f. Save the configurations.

11. Download the manifest.

    - To download the Microsoft Teams-specific manifest, launch the following URL:

      ```
      https://<ingress>/SmartViewM365/manifests/download/teams/
      smartviewm365manifest.zip
      ```

    - To download the Microsoft SharePoint-specific manifest, launch the following URL:

      ```
      https://<ingress>/SmartViewM365/manifests/download/sharepoint/
      smartviewm365manifest.zip
      ```

    For more information about uploading the manifest for Microsoft Teams and Microsoft SharePoint, see *OpenText Documentum Content Management for Microsoft 365 - Deployment and Administration Guide (EEMSODC250400-IGD)*.

3.6.20 Configuring Online Editing Service in Admin Console

To make the Microsoft 365 editing and co-authoring feature available in Admin Console, do the following:

If Admin Console is not deployed

1. In the `documentum\config\configuration.yml` file, add the following parameters in the `adminconsole.restApiRuntime.rest-api-runtime.properties` section:

   ```
   rest.security.headers.x_frame_options.disabled=false
   rest.security.headers.x_frame_options.policy=SAMEORIGIN
   ```

2. Save the changes.

If Admin Console is deployed

1. Run the following command to edit the ConfigMap of `adminconsole-rest-api-runtime-properties`.

   ```
   kubectl edit cm adminconsole-rest-api-runtime-properties
   ```

2. Add the following entries:

```
rest.security.headers.x_frame_options.disabled=false
rest.security.headers.x_frame_options.policy=SAMEORIGIN
```

3. Save the changes.

4. Restart the Admin Console pod.

3.6.21 Configuring OTDS account for Notification Service

Prerequisite

- Ensure that the OTDS service is deployed and working properly.

To create a system account user in OTDS for Notification Service:

1. Create a separate OTDS resources for each OpenText Documentum CM repository. For more information, see, *OpenText Directory Services - Installation and Administration Guide (OTDS250400-IWC)*.

2. Create an individual OTDS partitions corresponding to each repository. For more information, see, *OpenText Directory Services - Installation and Administration Guide (OTDS250400-IWC)*.

3. Upload the license to OTDS and apply it to all repositories on the OpenText Documentum CM. For more information, see "Licensing OpenText Documentum CM" on page 95.

4. Create a dedicated partition named `M365SystemPartition` in OTDS for the `OES connector service`.

   Note: You can use same partition created for OpenText Documentum CM for Microsoft 365.

5. Configure OTDS authentication:

   - Create a user in OTDS with the username `M365_SERVICE` and password `Xecmserviceuser@123` (password is case sensitive).
   - In the **Password Options** area:
     – Select **Do not require password change on reset**.
     – Clear the **User cannot change password** check box.
     – Select the **Password never expires** check box.
   - Create the user `M365_SERVICE` in the `M365SystemPartition`.
   - Allocate the license for your product. For more information, see *OpenText Directory Services - Installation and Administration Guide (OTDS250400-IWC)*.

   Note: Since `M365_SERVICE` is a service account, its password is automatically updated by the system after the service starts.

6. Consolidate the `M365_SERVICE` user to all required repositories. Ensure that the `M365_SERVICE` user is synced to each consolidated repositories.

7. Grant superuser privileges to the `M365_SERVICE` user in all available repositories.

8. Create an OTDS Client ID, for example, `oes_oauth_client`.

   - Ensure the **Confidential** option is not selected.
   - In the **Redirect URLs** tab, add the following:
     - `<Ingress URL>oes-connector`

9. Restart the notification service pod, after `M365_SERVICE` user is synchronized with the repository.

3.6.22 Configuring SMTP for client

Before configuring the SMTP properties, you must configure the mail server in client configuration. To configure the SMTP properties, you must modify the `smtp_c6.properties` variable in the `documentum/config/configuration.yml` file.

The following table lists the SMTP properties:

Property	Description
`hostname`	Indicates the host name of the SMTP mail server. The value of this property replaces the server specified in **Tools > Email** in client configuration.
`port`	Indicates the port number of the SMTP mail server. The value of this property replaces the port specified in **Tools > Email** in client configuration.
`from`	Indicates the email address for the `mail.smtp.from` property under `smtp_c6.properties`. The value of this property replaces the client end-user email address listed in the `dm_user` table.
`starttls.enable`	When enabled and supported by the server, you can use the `STARTTLS` command to switch the connections to a TLS- protected connection before issuing any login commands. The default value is `false`.
`auth`	When the value is set to `true`, it attempts to authenticate a user using the AUTH command. The default value is `false`.
`socketFactory.port`	Indicates the port used to connect to the specified socket factory. It uses the default port if you do not set the value.

Property	Description
`socketFactory.class`	When you set the value, it indicates the name of a class that implements the `javax.net.SocketFactory` interface. You can use this class to create SMTP sockets.
`socketFactory.fallback`	If the value is set to `true`, failure to create a socket using the specified socket factory class causes the socket to be created using the `java.net.Socket` class. The default value is `true`.

3.6.23 Configuring Brava certificate for client

1. Place the Brava certificate (self-signed or CA certificate provided by Brava for communication) in the following folders:
   - `d2classic/certificates/`
   - `d2smartview/certificates/`
2. In the `documentum/config/configuration.yml` file, set the value of the `bravatls.enable` variable to `true` and provide a value for the `bravatls.certName` variable.

For example:

```
bravatls:
    enable: true
    certName: <Certificate placed in d2classic/certificates/ and d2smartview/
certificates/>
```

3.6.24 Deploying customizations

3.6.24.1 Customizing webapp containers using single init container

For customizing webapps, you must build a container image where you can add JAR files, custom WAR files, plug-ins, and the property files.

For example, all the JAR or plug-in files must be placed in the `D2Customizations/<webapp_name>/D2Custom/WEB-INF/lib` folder and all the JavaScript files must be placed in the `D2Customizations/<webapp_name>/D2Custom/js` folder and so on.

Use the following common format for all client webapps where you can place all the custom files in their respective locations:

```
D2Customizations/
  D2Classic/
    D2Custom/
      js/               --> All D2Classic javascript files
      css/              --> All D2Classic css files
      wars/             --> External WAR files
      WEB-INF/lib       --> All D2Classic Plugins and Lib files
      WEB-INF/classes   --> All D2Classic properties files
      LP-Package/       --> All D2Classic language pack files
      D2-Ext/
```

```
        Widget1/        --> External Widget files
  D2Smartview/
    D2Custom/
      wars/             --> External WAR files
      WEB-INF/lib       --> All D2Smartview Plugins and Lib files
      WEB-INF/classes   --> All D2Smartview properties files
      LP-Package/       --> All D2Smartview language pack files
  D2Config/
    D2Custom/
      wars/             --> External WAR files
      WEB-INF/lib       --> All D2Config Plugins and Lib files
      LP-Package/       --> All D2Config language pack files
        sampleconfig1.zip   --> Custom import zip/xml file1
        sampleconfig2.zip   --> Custom import zip/xml file2
  D2Rest/
    D2Custom/
      wars/             --> External WAR files
      WEB-INF/lib       --> All D2Rest Plugins and Lib files
      LP-Package/       --> All D2Rest language pack files
AdminConsole/
    LP-Package/         --> All AdminConsole language pack files
```

Note: Smart View does not support Cascading Style Sheets (CSS), JavaScript (JS) or external widget customizations. If the new WAR files are added to the existing client-based pods, update the resources accordingly.

To build the container image:

1. Download the latest Alpine image from the repository.
2. To download the image to the local environment, copy the following content to a file named `Dockerfile` and update the Alpine image path based on your repository location:

   ```
   FROM <repository ip>:<port>/alpine:<image tag>
   ARG D2CUSTOM
   ENV D2_CUSTOM_PATH /customdir
   RUN adduser -D -H dmadmin && \
   mkdir -p $D2_CUSTOM_PATH && \
   chown -R dmadmin:dmadmin $D2_CUSTOM_PATH
   COPY --chown=dmadmin:dmadmin $D2CUSTOM/ $D2_CUSTOM_PATH/
   CMD sh
   ```

   The preceding contents is an example of a Docker file that can be used to build the container image.
3. Run the following command in the location where Docker file is created to build the container image:

   ```
   docker build -f Dockerfile -t <repository>/<custom_image_name>:<custom_image_tag> --
   build-arg D2CUSTOM=<D2customizations Dir> --no-cache .
   ```

   For example:

   ```
   docker build -f Dockerfile -t repo.opentext.com/d2webappcustomimage:23.2 --build-
   arg D2CUSTOM=D2Customizations --no-cache .
   ```
4. After the container image is created, do the following changes in the Helm charts when you deploy client:

   a. In the `documentum/dockerimages-values.yaml` file, add or update the `extraInitContainer` sections of the webapps as follows:

```
extraInitContainers: |
  - name: <init_container_name>
    image: <repository_path>/<init_image_name>:<image_tag>
    imagePullPolicy: Always
    command: ['/bin/sh', '-c', 'yes|cp -rf /customdir/<webapp_name>/* /opt/D2-
install/custom']
    volumeMounts:
      - name: customconfig
        mountPath: /opt/D2-install/custom
```

b. Update the image details with the container image and replace `<webapp_name>` with `D2Classic/D2Config/D2Smartview/D2Rest` based on the webapp that you plan to update in the `extraInitContainer` section.

   For example, the init container details for `D2Classic`:

```
extraInitContainers: |
  - name: customwebappinit
    image: repo.opentext.com/d2webappcustomimage:23.2
    imagePullPolicy: Always
    command: ['/bin/sh', '-c', 'yes | cp -rf /customdir/D2Classic/* /opt/D2-
install/custom']
    volumeMounts:
    - name: customconfig
      mountPath: /opt/D2-install/custom
```

c. To enable language pack for Admin Console, add the following init container details:

```
extraInitContainers: |
  - name: customwebappinit
    image: <remote-repository>/<init-image-name>:<init-image-tag>
    imagePullPolicy: Always
    command: ['/bin/sh', '-c', 'yes | cp -rf /customdir/AdminConsole/D2Custom/
* /opt/D2-install/custom']
    volumeMounts:
    - name: customconfig
      mountPath: /opt/D2-install/custom
```

d. In the `documentum/config/configuration.yml` file, set the `custom` variable to `true` for the respective webapps:

```
customConfigurations:
  custom: true
```

5. In the client configuration webapp:

   a. If any plug-in JAR files are added to the customization, you must add the corresponding plug-in JAR file names in the `d2config plugins` section in the `documentum/config/configuration.yml` file.

   For example:

```
d2config:
  customConfigurations:
    custom: true
    - Plugin1
    - Plugin2
```

   b. A properties file with customized attributes can be added directly to the d2config subchart in `documentum/charts/d2config/properties` before the deployment. Provide the file name in the `customConfigurations` section of `d2config` in the `documentum/config/configuration.yml` file.

   For example:

```
d2config:
  customConfigurations:
    custom: true
    settings: properties/D2Config.properties
```

Note: If both the custom plug-in JAR files and the custom `D2Config.properties` files are used in the single deployment, the custom `D2Config.properties` file overrides the plug-in JAR names.

c. To import multiple custom client configuration files, place the custom files in the `D2Customizations/D2Config/<custom_files>` location. Then, update the following values in the `documentum/config/configuration.yml` file:

```
d2config:
  customConfigurations:
    custom: true
    filename: <custom_configuration_file_names_with_extension_comma_separated
e.g., D2_Test_Config1.zip,D2_Test_Config2.xml>
```

Notes

- Custom import supports only the XML or ZIP formats.
- Observe the client configuration pod logs for successful completion of multiple config imports (or for any errors) before accessing the client configuration pod.

6. If the WAR file customization is used, uncomment the following section in the `documentum/config/configuration.yml` file and update `serviceName` with the custom webapp name:

```


#extraPaths is used to define additional ingress resources for custom wars.
#For aws path should be as follows path: /<path>*extraPaths:
extraPaths:
  - backend:
       serviceName: <servicename>
       servicePort: 8080
     path: /<path>
```

3.6.24.2 Customizing webapp containers using hook scripts

To customize webapps using the hook script, you need to create a container image with scripts and the necessary files in it. Place the hook script files according to the structure as follows:

```
D2Customizations/
  D2Classic/
    hook_scripts(*deploy.sh)  --> hook scripts of Classic View webapp
    applicable_files          --> Necessary files/dirs/sub-dirs referred in the hook
                                  script
  D2Smartview/
    hook_scripts(*deploy.sh)  --> hook scripts of Smart View and necessary files
    applicable_files          --> Necessary files/dirs/sub-dirs referred in the hook
                                  script
  D2Config/
    hook_scripts(*deploy.sh)  --> hook scripts of client configuration and necessary
files
    applicable_files          --> Necessary files/dirs/sub-dirs referred in the hook
                                  script
  D2Rest/
```

```
    hook_scripts(*deploy.sh)  --> hook scripts of Client REST API and necessary files
    applicable_files          --> Necessary files/dirs/sub-dirs referred in the hook
                                  script
```

Note: The name of the hook scripts can be preceded by the execution order number, for example, `10deploy.sh` runs first before `20deploy.sh` and then `30deploy.sh` etc.

1. Copy the following content to a file and name it `Dockerfile`. Modify the Alpine image path based on your repository location to download the image to the local environment.

   ```
   FROM <repository_ip>:<port>/alpine:<image_tag>
   ARG D2CUSTOM

   ENV D2_CUSTOM_PATH /customdir

   RUN adduser -D -H dmadmin && \
       mkdir -p $D2_CUSTOM_PATH && \
       chown -R dmadmin:dmadmin $D2_CUSTOM_PATH

   COPY --chown=dmadmin:dmadmin $D2CUSTOM/ $D2_CUSTOM_PATH/
   CMD sh
   ```

   The preceding content is an example of the Docker file you can use to build the image.

2. Run the following command in the location where `Dockerfile` is created and the necessary script files are present to build the container image.

   ```
   docker build -f Dockerfile -t <repository>/custom_image_name>:<custom_image_tag> --
   build-arg D2CUSTOM=<D2Scripts Dir> -- no cache .
   ```

   For example:

   ```
   docker build -f Dockerfile -t repo.opentext.com/d2webapphook:23.2 --build-arg
   D2CUSTOM=D2Customizations --no-cache .
   ```

3. After the container image is generated, make the following change in the Helm charts when deploying client:

   - In the `dockerimages-values.yaml` file, add or update the `extraInitContainers` sections of the respective webapp:

     ```
     extraInitContainers: |
       - name: <init_container_name>
         image: <repository_path>/<init_image_name>:<image_tag>
         imagePullPolicy: Always
         command: ['/bin/sh', '-c', 'yes | cp -rf /customdir/<webapp_name>/* /opt/
     D2-install/custom/']
         volumeMounts:
         - name: customconfig
           mountPath: /opt/D2-install/custom
           subPath: <webapp_service_name>
     ```

     For example, update image details with the container image created previously and replace `<webapp_name>` with `D2Classic/D2Config/D2Smartview/D2Rest` and `<webapp_service_name>` with `d2classic/d2config/d2smartview/d2rest` based on the webapp you want to update in the `extraInitContainers` section.

```
extraInitContainers: |
  - name: customwebappinit
    image: repo.opentext.com/d2webapphook:23.2
    imagePullPolicy: Always
    command: ['/bin/sh', '-c', 'yes | cp -rf /customdir/D2Classic/* /opt/D2-
install/custom/']
    volumeMounts:
    - name: customconfig
      mountPath: /opt/D2-install/custom
      subPath: d2classic
```

- In the `documentum/values.yaml` file (or `configuration.yml` if used), set `custom`, `hook_approach`, and `createPVC` to `true` in the `customConfigurations` section for the respective webapps.

```
customConfigurations:
  custom: true
  hook_approach: true
  createPVC: true
```

Notes

– If hook approach customization is used for multiple webapps, make sure `createPVC` is set to `true` only for one of the webapps and `false` for the remaining webapps.

– `/opt/D2CS-install` is the PVC mount location of `customscriptpvc` in the Documentum CM Server pod.

– If Reports and `hook_approach` are enabled, add `d2config` as a subpath value in the `d2config.extraInitContainers.volumeMounts.subPath` variable.

3.6.24.3 Customizing webapps containers using single init container including hook scripts (hybrid approach)

Use the following structure to build the container image:

```
D2Customizations/
  D2Classic/
    hook_scripts(*deploy.sh)   --> hook scripts of d2 classic and necessary files
    applicable_files           --> Necessary files/dirs/sub-dirs referred in the hook
                                   script
    D2Custom/
      js/                      --> All D2Classic javascript files
      css/                     --> All D2Classic css files
      wars/                    --> External WAR files
      WEB-INF/lib              --> All D2Classic Plugins and Lib files
      WEB-INF/classes          --> All D2Classic properties files
      LP-Package/              --> All D2Classic language pack files
      D2-Ext/
        Widget1/               --> External Widget files
  D2Smartview/
    hook_scripts(*deploy.sh)   --> hook scripts of d2 smartview and necessary files
    applicable_files           --> Necessary files/dirs/sub-dirs referred in the hook
                                   script
    D2Custom/
      wars/                    --> External WAR files
      WEB-INF/lib              --> All D2Smartview Plugins and Lib files
      WEB-INF/classes          --> All D2Smartview properties files
      LP-Package/              --> All D2Smartview language pack files
  D2Config/
    hook_scripts(*deploy.sh)   --> hook scripts of d2 config and necessary files
    applicable_files           --> Necessary files/dirs/sub-dirs referred in the hook
                                   script
    D2Custom/
      wars/                    --> External WAR files
      WEB-INF/lib              --> All D2Config Plugins and Lib files
      LP-Package/              --> All D2Config language pack files
        sampleconfig1.zip      --> Custom import zip/xml file1
        sampleconfig2.zip      --> Custom import zip/xml file2
  D2Rest/
    hook_scripts(*deploy.sh)   --> hook scripts of d2 rest and necessary files
    applicable_files           --> Necessary files/dirs/sub-dirs referred in the hook
                                   script
    D2Custom/
      wars/                    --> External WAR files
      WEB-INF/lib              --> All D2Rest Plugins and Lib files
      LP-Package/              --> All D2Rest language pack files
```

Follow the steps shown in "Customizing webapp containers using single init container" on page 137 and "Customizing webapp containers using hook scripts" on page 140 to update the `documentum/config/configuration.yml` files with respect to the webapp used.

Note: Make sure `subPath:` is added for the init container of the respective webapp while using the hybrid approach.

3.6.24.4 Language pack support

For a list of available language support packs, see the product *Release Notes*. This list also includes the language codes required for deployment.

You can install language pack without customization by adding the required language pack in the `global.locale` variable in the `documentum/values.yaml`. You can add multiple language packs separated by comma.

For example:

```
global:
  locale:fr,en,es
```

3.6.24.4.1 Installing language pack through customization

To implement a language pack, build a container image with the structure as described in "Customizing webapp containers using single init container" on page 137 and place all of the required language pack files in the **LP-Package** folder of each webapp. The following are the required files:

- AdvancedPublishing-LanguagePack_xx.zip
- C2-LanguagePack_xx.zip
- D2-Bin-LanguagePack_xx.zip
- D2-LanguagePack_xx.zip
- D2-Mobile_LanguagePack_xx.zip
- D2-Smartview_LanguagePack_xx.zip
- D2-Config-LanguagePack_xx.zip

- AdminConsole-LanguagePack_xx.zip

  Note: client configuration supports only French and English language packs.

Where `xx` is the language code. For example, the French language pack name would be `D2–LanguagePack_fr.zip`.

1. After the image is built, update the `extraInitContainers` section of each webapp in the `documentum/dockerimages-values.yaml` file as described in "Customizing webapp containers using single init container" on page 137.
2. In the `documentum/values.yaml` file, update the `global.locale` variable with the required language code which will be used by Documentum CM Server and all the client webapps. If a specific webapp requires a different locale code, update the locale code for that webapp in the `documentum/config/configuration.yml` file.

   - Example 1: Add the French language code for Classic View in the `documentum/config/configuration.yml` file:

     ```
     d2classic
       locale:fr
     ```

   - Example 2: If the language code updated in `global.locale` variable is `fr`, you can add an additional language code for Classic View in the `documentum/config/configuration.yml` file:

     ```
     global:
       locale:fr

     d2classic
       locale:fr,es
     ```

   Note: For client webapps, the locale codes specified in the `documentum/config/configuration.yml` file takes precedence when values are provided in both `global.locale` variable and `configuration.yml` file.

3. Make the following changes in the browser to view the client in the respective languages (applicable to the Google Chrome browser):

   a. Go to **Settings > Advanced > Languages > Add Language**.
   b. Search for the required language and click **Add**.
   c. Click the ellipses (...) menu of the added language and select **Move to the top**.
   d. Relaunch the browser and access client.

3.6.25 Configuring Reports users in Documentum Administrator to access reports

After the successful deployment of client pod and enabling the pod with Reports, configure the Reports users in Documentum Administrator to access the reports.

To view reports:

1. Log in to Documentum Administrator.
2. Open **Administration** > **User Management** > **Groups** > **dctm-reports-users**.
3. Go to **File** > **Add members**.
4. Add the newly created or existing user.
5. Log in to client with the user credentials and access the reports.

To create or edit report templates:

1. Open **Administration** > **User Management** > **Users**.
2. Search the user, right-click, and select **Assign Group Membership**.
3. Search for **dctm-reports-designer** and add it to the selected list using the right arrow.
4. Search for **d2-admins**, add it to the selected list, and click **OK**.
5. Log in to the client with the user credentials and access the reports.

3.6.26 Configuring Reports options in client configuration for Smart View

1. Log in to the client configuration using the administrative account.
2. Select **Tools** > **Options** > **DCTM Reports Options** and provide the Reports configuration details such as:
   - **Core Application URL**: Reports core application path.

     For example: `https://<fully qualified ingress host>/dtr`.
   - **Date Pattern**: `M/dd/yyyy hh:mm:ss a`
   - **Framework**: The value is `HTML5`.
   - **Preview Inline**: You can enable this option to view the Reports in the Smart View context.
3. Click **Tools** > **Refresh Cache** to refresh the configurations.

3.6.27 Deploying add-on components

When using `configuration.yml` and the add-ons configuration YAML files (for example, `ao-config.yaml`) during Helm deployment or upgrade, make sure to include all the file name values of `configuration.yml` in the add-ons configuration file. For example, if the `d2config` section in `configuration.yml` has the following file name values:

```
d2config:
 customConfigurations:
  filename:
  - <docbase_name>:DCTM-Reports-Application-25.4.0-Export-Config.zip
```

You must add the same file name values (`<docbase_name>:DCTM-Reports-Application-25.4.0-Export-Config.zip`) in the configuration file of add-ons such as the `ao-config.yaml` file, along with the other client configurations.

```
d2config:
  customConfigurations:
    filename:
    - <docbase_name>:OpenText_EPFM_Asset_Operations_Export_Config.zip
    - <docbase_name>:DCTM-Reports-Application-25.4.0-Export-Config.zip
```

3.6.28 Setting up Google Kubernetes Engine (GKE) native load balancer for client

The Google Kubernetes Engine (GKE) native load balancer is secure and employs an efficient gateway for load balancing. Configure the GKE native load balancer when deploying OpenText Documentum CM solutions to Google Cloud Platform.

3.6.28.1 Prerequisites

- Create an SSL policy using the following command:

  ```
  gcloud compute ssl-policies create <ssl-policy-name> --profile COMPATIBLE --min-tls-
  version 1.0
  ```

- Create a Kubernetes TLS Secret certificate using the following command:

  ```
  kubectl create secret tls <secret-name> --namespace <namespace name> --key <private-
  key-file-name with .key extension> --cert <cert-file-name with .cert extension>
  ```

- Create an external static IP address in Google Cloud Platform. For more information, see the *Configure static external IP addresses* topic in the *Google Compute Engine* documentation.

3.6.28.2 Configuring native load balancer for GKE Classic View, client configuration, Smart View, and Client REST API

After you have deployed OpenText Documentum CM, perform the following steps to configure the native load balancer for GKE.

To configure the native load balancer for GKE:

1. Create a `FrontendConfig` Custom Resource Definition (CRD) to define the SSL policy and redirections from HTTP to HTTPS. To create a frontend object, you can see the following sample code.

```
apiVersion: networking.gke.io/v1beta1
kind: FrontendConfig
metadata:
  name: frontend-config
spec:
  sslPolicy: <ssl-policy name> #ssl-policy created as part of the pre-requisite
above
  redirectToHttps:
    enabled: true
    responseCodeName: MOVED_PERMANENTLY_DEFAULT
```

2. Create a Classic View backend configuration YAML file.

```
apiVersion: cloud.google.com/v1
kind: BackendConfig
metadata:
  name: {{ customer }}-{{ env }}-classic-backend
spec:
  sessionAffinity:
    affinityType: "GENERATED_COOKIE"
    affinityCookieTtlSec: 50
  timeoutSec: 180
  connectionDraining:
    drainingTimeoutSec: 600
  logging:
    enable: false
    sampleRate: 0.5
  healthCheck:
    checkIntervalSec: 8
    timeoutSec: 5
    healthyThreshold: 2
    unhealthyThreshold: 3
    type: HTTP
    requestPath: /D2/ready
    port: 8080
```

3. Create a client configuration backend configuration YAML file.

```
apiVersion: cloud.google.com/v1
kind: BackendConfig
metadata:
  name: {{ customer }}-{{ env }}-config-backend
spec:
  sessionAffinity:
    affinityType: "GENERATED_COOKIE"
    affinityCookieTtlSec: 50
  timeoutSec: 180
  connectionDraining:
    drainingTimeoutSec: 600
  logging:
    enable: false
    sampleRate: 0.5
  healthCheck:
    checkIntervalSec: 8
    timeoutSec: 5
    healthyThreshold: 2
    unhealthyThreshold: 3
    type: HTTP
    requestPath: /D2-Config/ready
    port: 8080
```

4. Create a Smart View backend configuration YAML file.

```
apiVersion: cloud.google.com/v1
kind: BackendConfig
metadata:
  name: {{ customer }}-{{ env }}-smartview-backend
spec:
  sessionAffinity:
    affinityType: "GENERATED_COOKIE"
    affinityCookieTtlSec: 50
  timeoutSec: 180
  connectionDraining:
    drainingTimeoutSec: 600
  logging:
    enable: false
    sampleRate: 0.5
  healthCheck:
    checkIntervalSec: 8
    timeoutSec: 5
    healthyThreshold: 2
    unhealthyThreshold: 3
    type: HTTP
    requestPath: /D2-Smartview/services
    port: 8080
```

5. Create a Client REST API backend configuration YAML file.

```
apiVersion: cloud.google.com/v1
kind: BackendConfig
metadata:
  name: {{ customer }}-{{ env }}-rest-backend
spec:
  sessionAffinity:
    affinityType: "GENERATED_COOKIE"
    affinityCookieTtlSec: 50
  timeoutSec: 180
  connectionDraining:
    drainingTimeoutSec: 600
  logging:
    enable: false
    sampleRate: 0.5
  healthCheck:
    checkIntervalSec: 8
    timeoutSec: 5
    healthyThreshold: 2
    unhealthyThreshold: 3
    type: HTTP
    requestPath: /D2-rest/services
    port: 8080
```

6. Run the following command to apply all the backend config YAML files to Kubernetes:

```
kubectl apply -f <backend config yaml file>
```

7. Check the status of all the backend configuration YAML files.

```
kubectl get backendconfig
```

8. Create the following patch YAML files for client webapps with annotations for backend configurations and Network Endpoint Groups (NEG).

- Create a Classic View patch YAML.

  ```
  metadata:
    annotations:
      cloud.google.com/backend-config: '{"ports": {"8080":"<Classic View
  backendconfig name>"}}'
      cloud.google.com/neg: '{"ingress": true}'
  ```

- Create a client configuration patch YAML file.

  ```
  metadata:
    annotations:
      cloud.google.com/backend-config: '{"ports": {"8080":"<client configuration
  backendconfig name>"}}'
      cloud.google.com/neg: '{"ingress": true}'
  ```

- Create a Smart View patch YAML file.

  ```
  metadata:
    annotations:
      cloud.google.com/backend-config: '{"ports": {"8080":"<Smart View
  backendconfig name>"}}'
      cloud.google.com/neg: '{"ingress": true}'
  ```

- Create a Client REST API patch YAML file.

  ```
  metadata:
    annotations:
      cloud.google.com/backend-config: '{"ports": {"8080":"<Client REST API
  backendconfig name>"}}'
      cloud.google.com/neg: '{"ingress": true}'
  ```

9. Run the following command to patch the services annotations:

   ```
   kubectl patch svc <service name> --type merge --patch-file <patch yaml file> -n
   patch yaml file> -n <namespace>
   ```

10. Run the following command to create an ingress resource referencing the `FrontendConfig` and static IP address:

    ```
    apiVersion: networking.k8s.io/v1
    kind: Ingress
    metadata:
      name: <ingress resource name>
      annotations:
        kubernetes.io/ingress.allow-http: "true"
        kubernetes.io/ingress.global-static-ip-name: <static IP name>
        networking.gke.io/v1beta1.FrontendConfig: <frontend config name>
    spec:
      tls:
      - hosts:
        - <ingress host name >
        secretName: <kubernetes secret name containing SSL certificate>
     ingressClassName: <leave blank>
      rules:
      - host: <ingress host name>
        http:
          paths:
          - pathType: Prefix
            backend:
              service:
                name: <service name>
                port:
                  number: <service port number>
            path: /<webapp endpoint>
    ```

11. Run the following command to create a GKE native load balancer. This process takes 15 to 20 minutes.

```
kubectl apply -f <ingres yaml file>
```

12. Ensure the following points to monitor the load balancer in GKE:

    - All the load balancer backends are pointing to the respective protocol, port, and request path as defined in the backend configuration files.
    - All the backends are responding correctly to health checks to maintain high availability.
    - The frontend configurations defined in `frontendconfig` include the SSL policy and the HTTP to HTTPS redirect configuration.

You can access all the application URLs using the browser to confirm if they are running.

3.6.29 Integrating Client Branch Office Caching Services with client

The client uses Client Branch Office Caching Services to communicate with one or more Branch Office Caching Services server or Accelerated Content Services that act as Branch Office Caching Services servers. Branch Office Caching Services servers improve file transfer performance by connecting to a local server, even if they are remotely located, allowing users to checkin, import, and request files using Branch Office Caching Services.

Client Branch Office Caching Services is not containerized. Therefore, you must deploy Client Branch Office Caching Services in an on-premises environment. For more information to deploy Client Branch Office Caching Services in an on-premises environment, see *OpenText Documentum Content Management - Client Installation Guide (EDCCL250400-IGD)*.

To integrate on-premises Client Branch Office Caching Services with OpenText Documentum CM cloud:

1. In the `d2config` section of the `configuration.yml` file, add the following properties to the custom `D2_Config_properties` file:

```
d2config:
  D2_Config_properties: |-
    D2-BOCS=true
```

2. In the `d2classic` or `d2smartview` section of the `configuration.yml` file, add the following properties to the custom `D2FS_properties` file:

```
d2classic:
  D2FS:
    D2FS_properties: |-
      pluginsOrder=D2-BOCS,C2,O2
      D2-BOCS=true

d2smartview:
  D2FS:
    D2FS_properties: |-
```

```
      pluginsOrder=D2-BOCS,C2,O2
      D2-BOCS=true
```

3. In the `d2classic` section of the `configuration.yml` file, add the following properties to the custom `settings_properties` file:

   ```
   d2classic:
     settings_properties: |-
       login.networklocation.hide = false
   ```

4. In `allowedFrameOrigins` of the`d2classic` or `d2smartview` section of the `configuration.yml` file, add the on-premises Client Branch Office Caching Services URL:

   ```
   d2classic:
     settings:
       allowedFrameOrigins: frame-ancestors http(s)://<Client Branch Office Caching
   Services URL>/*

   d2smartview:
     restApiRuntime:
       ContentConnect:
         restAllowedOrigins: frame-ancestors http(s)://<Client Branch Office Caching
   Services URL>/*
   ```

3.6.30 Modifying server.ini and server.xml files

OpenText Documentum CM supports externalizing Documentum CM Server capabilities configured in the `server.ini` file using a `ConfigMap` file.

Similarly, the `server.xml` file of Tomcat web application server is externalized using the global variables in the `documentum/values.yaml` file for Documentum CM Server, Independent Java Method Server, and all OpenText Documentum CM components.

Modifying `server.ini` of Documentum CM Server

You can modify the values only for the following keys:

- `autoRestart`

  Valid values are `true` and `false`.

- `concurrent_sessions`

  Valid value is a numerical value.

For example, if you want to modify the value of `concurrent_sessions` after the initial deployment, do the following:

1. Run the following command:

   ```
   kubectl edit cm cs-server-ini-configmap -n <namespace>
   ```

Note: If you do not want to modify the value of `autoRestart` and `concurrent_sessions` defined in the initial deployment, make sure that those values in the `cs-server-ini-configmap.yaml` file of the existing deployment is updated in the `documentum/charts/content-server/templates/cs-server-ini-configmap.yaml` file prior to the Helm upgrade process.

2. Run the Helm upgrade command.

Modifying `server.xml` of Tomcat for Documentum CM Server and Independent Java Method Server

You can modify the `server.xml` file of Tomcat web application server for both Documentum CM Server and Independent Java Method Server only for the following variables in the `documentum/values.yaml` file:

- `global.tomcatMaxThreads`
- `global.tomcatConnectionTimeout`

To provide the appropriate values, open the YAML file, and read the descriptions of the variables.

For example, if you want to modify the value of `global.tomcatMaxThreads` after the initial deployment, do the following:

1. Open the `documentum/values.yaml` file.
2. Modify the value of `global.tomcatMaxThreads`.
3. Run the Helm upgrade command.
4. (For Independent Java Method Server) Restart the Independent Java Method Server pod.
5. (For Documentum CM Server) Restart Java Method Server.

Modifying `server.xml` of Tomcat for all other components

You can modify the `server.xml` file of Tomcat web application server for all other components only for the following variables in the `documentum/values.yaml` file:

- `global.tomcatConnectionTimeout`
- `global.tomcatMaxHttpHeaderSize`
- `global.tomcatMaxThreads`
- `global.tomcatKeepAliveTimeout`
- `global.tomcatAcceptCount`

To provide the appropriate values, open the YAML file, and read the descriptions of the variables.

For example, if you want to modify the value of `global.tomcatMaxThreads` after the initial deployment, do the following:

1. Open the `documentum/values.yaml` file.
2. Modify the value of `global.tomcatMaxThreads`.
3. Run the Helm upgrade command.

Chapter 4

Using features in cloud platforms

This documentation provides information about the list of cloud features integrated with OpenText Documentum CM and the variables that you must update to use the features, if required for your environment.

4.1 Vault

OpenText Documentum CM supports the following vault types:

- HashiCorp Vault: Stores and retrieves all secrets automatically from the HashiCorp Vault server.

  The HashiCorp Vault type is supported on both the on-premises and cloud environments.
- Kubernetes native secrets: Retrieves all secrets from Kubernetes secrets.

  OpenText Documentum CM retrieves the passwords automatically from Kubernetes API instead of storing them as environment variables.

  The Kubernetes native secrets vault type is supported only on cloud environments.

Notes

- The Documentum Secret Integration Service (DSIS) daemon agent is started automatically at the time of the Documentum CM Server deployment.

  For more information about DSIS, see *OpenText Documentum Content Management - Server and Server Extensions Installation Guide (EDCSY250400-IGD)*.
- Provide the appropriate value for all the relevant vault type related variables in the `documentum/config/passwords_vault.yaml` file if for HashiCorp Vault and in the `documentum/config/passwords_k8api.yaml` file for Kubernetes native secrets only if you want to use a vault type.

To use HashiCorp Vault in OpenText Documentum CM deployment:

1. Configure the HashiCorp Vault server before deploying, upgrading, or migrating OpenText Documentum CM and make sure that the HashiCorp Vault server is accessible from the cluster.

   For more information about configuring the HashiCorp Vault server, see *HashiCorp* documentation.
2. Store all the secret ID information in the `<secret_name>|<key_name>` format in the HashiCorp Vault server.

For more information about storing the secret IDs, see *HashiCorp* documentation.

The list of secret and key names is available in the respective product *Installation Guide* or *Deployment Guide*.

Note: (Only for Documentum CM Server) If key name is the host name, make sure that you create keys for the primary pod and all remote Content Servers/High-availability pods. Make sure that you provide values for *`<namespace>`*, *`<docbase_name>`*, and so on in the `documentum/config/passwords_vault.yaml` file.

3. Provide the appropriate values for all variables to pass them to your templates. To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

4. After providing the appropriate values for all the HashiCorp Vault-related variables, deploy OpenText Documentum CM.

   - To use HashiCorp Vault during the initial deployment, run the Helm install command.
   - To use HashiCorp Vault after the initial deployment, run the Helm upgrade command.

   Note: Use `passwords_vault.yaml` in the `helm install` or `helm upgrade` command instead of `passwords.yaml`.

5. Run the following command to verify the pod deployment status:

```
kubectl describe pods <name of the pod>
```

To use Kubernetes native secrets in OpenText Documentum CM deployment:

1. Create the Kubernetes secret before deploying, upgrading, or migrating as OpenText Documentum CM Helm charts depend on this secret.

   Run the following command to create a Kubernetes secret:

```
kubectl apply -f vault_secret.yaml
```

2. Update password values for all the appropriate variables in the `documentum/config/vault_secret.yaml` file.

3. Provide the appropriate values for all variables to pass them to your templates. To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

4. After providing the appropriate values for all the Kubernetes native secret-related variables, deploy OpenText Documentum CM.

   - To use Kubernetes native secrets during the initial deployment, run the Helm install command.
   - To use Kubernetes native secrets after the initial deployment, run the Helm upgrade command.

Note: Use `passwords_k8api.yaml` in the `helm install` or `helm upgrade` command instead of `passwords.yaml`.

5. Run the following command to verify the pod deployment status:

```
kubectl describe pods <name of the pod>
```

Global vault type variables for all components

Vault type variables in the documentum/config/passwords_vault file

- `installOwnerPassword`
- `otdsAdminPassword`
- `databaseAdminPassword`
- `globalRegistryPassword`
- `trustStorePassword`

Vault type variables in the documentum/config/passwords_k8api file

- `installOwnerPassword`
- `otdsAdminPassword`
- `databaseAdminPassword`
- `globalRegistryPassword`
- `trustStorePassword`
- `otdsClientSecret`

Vault type variables in the documentum/values.yaml file

- `global.isVaultEnabled`
- `global.vaultType`
- `global.secretConfigName`
- `global.aekLocation`

Note: Provide a valid value for `global.secretConfigName` only if you want to use Kubernetes native secrets.

Vault type variables for Documentum CM Server

Vault type variables in the documentum/config/passwords_vault.yaml file

- `cs-secrets.docbase.password`
- `cs-secrets.docbroker.certificate.password`
- `cs-secrets.docbroker.certificate.aekpassphrase`
- `cs-secrets.docbroker.certificate.trustpassword`

- `cs-secrets.contentserver.installOwner.password`
- `cs-secrets.contentserver.globalRegistry.password`
- `cs-secrets.contentserver.aek.passphrase`
- `cs-secrets.contentserver.certificate.password`
- `cs-secrets.contentserver.certificate.trustpassword`
- `cs-secrets.contentserver.install.root.password`
- `cs-secrets.database.password`
- `cs-secrets.email.smtpPass`
- `cs-logging-configMap.fluentdConf.kafkaUsrPasswd`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `cs-secrets.docbase.password`
- `cs-secrets.docbroker.certificate.password`
- `cs-secrets.docbroker.certificate.aekpassphrase`
- `cs-secrets.docbroker.certificate.trustpassword`
- `cs-secrets.contentserver.installOwner.password`
- `cs-secrets.contentserver.globalRegistry.password`
- `cs-secrets.contentserver.aek.passphrase`
- `cs-secrets.contentserver.certificate.password`
- `cs-secrets.contentserver.certificate.trustpassword`
- `cs-secrets.contentserver.install.root.password`
- `cs-secrets.database.password`
- `cs-secrets.email.smtpPass`
- `cs-secrets.ingress.keystorepassword`
- `cs-logging-configMap.fluentdConf.kafkaUsrPasswd`

Vault type variables in the documentum/config/configuration.yml file

- `docbroker.vault.existVolumePv`
- `content-server.vault.existVolumePv`
- `content-server.vault.dsisCreatePVC`

Note: Make sure that you create a secret ID in the HashiCorp Vault server in the `PRIMARY_DOCBASE_INSTALL_OWNER_PASSWORD/`*`<repository name>`* format.

Vault type variables for client

Vault type variable in the documentum/config/passwords_vault.yaml file

- `d2rest.msgraphConfig.clientsecret`

Vault type variable in the documentum/config/passwords_k8api.yaml file

- `d2rest.msgraphConfig.clientsecret`

Vault type variables in the documentum/config/configuration.yml file

- `d2config.vault.statusCheckRetryCount`
- `d2config.vault.statusCheckTimeInterval`
- `d2classic.vault.statusCheckRetryCount`
- `d2classic.vault.statusCheckTimeInterval`
- `d2smartview.vault.statusCheckRetryCount`
- `d2smartview.vault.statusCheckTimeInterval`
- `d2rest.vault.statusCheckRetryCount`
- `d2rest.vault.statusCheckTimeInterval`

Vault type variables for OpenText AppWorks Gateway

Vault type variable in the documentum/config/passwords_vault.yaml file

- `appworks-gateway.newrelic.licenseKey`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `appworks-gateway.database.admin.password`
- `appworks-gateway.database.appworksdb.password`
- `appworks-gateway.otds.admin.password`
- `appworks-gateway.awg.admin.newadminpassword`

Vault type variable in the documentum/config/configuration.yml file

- `appworks-gateway.vault.type`
- `appworks-gateway.vault.host`
- `appworks-gateway.vault.port`
- `appworks-gateway.vault.namespace`
- `appworks-gateway.vault.secrets`
- `appworks-gateway.vault.authConfigPath`
- `appworks-gateway.vault.authPath`

- `appworks-gateway.vault.role`
- `appworks-gateway.vault.authType`

Vault type variable in the documentum/values.yaml file

- `appworks-gateway.vault.enabled`
- `appworks-gateway.vault.serviceAccount`

Vault type variables for OpenText Intelligent Viewing

Vault type variables in the documentum/config/configuration.yml file

- `otiv.global.secretlink.vault.address`
- `otiv.global.secretlink.vault.mountpoint`
- `otiv.global.secretlink.vault.path`
- `otiv.global.secretlink.vault.namespace`
- `otiv.global.secretlink.vault.authpath`
- `otiv.global.secretlink.vault.role`

Vault type variable in the documentum/values.yaml file

- `otiv.global.secretlink.enabled`

Vault type variables for Admin Console

Vault type variables in the documentum/config/configuration.yml file

- `adminconsole.vault.statusCheckRetryCount`
- `adminconsole.vault.statusCheckTimeInterval`

Vault type variables for Foundation SOAP API

Vault type variable in the documentum/config/passwords_vault.yaml file

- `dfs.dfc.globalRegistryPassword`

Vault type variable in the documentum/config/passwords_k8api.yaml file

- `dfs.dfc.globalRegistryPassword`

Vault type variables for Foundation REST API

Vault type variables in the documentum/config/configuration.yml file

- `dctm-rest.vault.statusCheckRetryCount`
- `dctm-rest.vault.statusCheckTimeInterval`

Vault type variables for Documentum Administrator

Vault type variables in the documentum/config/passwords_vault.yaml file

- `installOwnerPassword`
- `globalRegistryPassword`
- `cs-secrets.clients.preferencePassword`
- `cs-secrets.client.presetPassword`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `installOwnerPassword`
- `globalRegistryPassword`
- `cs-secrets.clients.preferencePassword`
- `cs-secrets.client.presetPassword`

Vault type variables in the documentum/config/configuration.yml file

- `da.vault.size`
- `da.vault.createPVC`
- `da.vault.existVolumePv`

Vault type variables for Records Client

Records Manager

Vault type variables in the documentum/config/passwords_vault.yaml file

- `installOwnerPassword`
- `globalRegistryPassword`
- `cs-secrets.clients.preferencePassword`
- `cs-secrets.client.presetPassword`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `installOwnerPassword`
- `globalRegistryPassword`
- `cs-secrets.clients.preferencePassword`
- `cs-secrets.client.presetPassword`

Vault type variables in the documentum/config/configuration.yml file

- `records.vault.createPVC`

- `records.vault.size`
- `records.vault.existVolumePv`

Records Queue Manager

Vault type variables in the documentum/config/passwords_vault.yaml file

- `installOwnerPassword`
- `globalRegistryPassword`
- `cs-secrets.clients.preferencePassword`
- `cs-secrets.client.presetPassword`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `installOwnerPassword`
- `globalRegistryPassword`
- `cs-secrets.clients.preferencePassword`
- `cs-secrets.client.presetPassword`

Vault type variables in the documentum/config/configuration.yml file

- `rqm.vault.createPVC`
- `rqm.vault.size`
- `rqm.vault.existVolumePv`

Vault type variables for Documentum xPlore

Vault type variables in the documentum/config/passwords_vault.yaml file

- `xPlore-OneD.cps.extraEnv.ESS_PASSWORD`
- `xPlore-OneD.indexserver.extraEnv.PASSWORD`

Vault type variable in the documentum/config/passwords_k8api.yaml file

- `xPlore-OneD.cps.extraEnv.ESS_PASSWORD`
- `xPlore-OneD.indexserver.extraEnv.PASSWORD`

Vault type variables for Independent Java Method Server

Vault type variable in the documentum/config/passwords_vault.yaml file

- `ijms.globalRegistryPassword`

Vault type variable in the documentum/config/passwords_k8api.yaml file

- `ijms.globalRegistryPassword`

Note: Add the Independent Java Method Server application server password secret name and key name in the *<secret_name>/<key_name>* format in the HashiCorp Vault server. The secret name is `DCTM_SERVER_JMX_PASSWORD` and the key name is *<Independent Java Method Server host name>*.

Repeat this for all replica pods.

Vault type variables for OpenText Content Aviator

Vault type variables in the documentum/config/passwords_vault.yaml file

- `dctm-dcis.docbasePassword`
- `dctm-dcis.dfc.globalRegistryPassword`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `dctm-dcis.docbasePassword`
- `dctm-dcis.dfc.globalRegistryPassword`

Notes

- Store the `ACTIVEMQ_PASSWORD` with secret key name `ACTIVEMQ_PASSWORD/dcis` in the HashiCorp Vault server.
- Store the `OAUTH_CLIENT_SECRET` with secret key name `OAUTH_CLIENT_SECRET/dcis` in the HashiCorp Vault server.
- Store the `DOCBASE_PASSWORD` with secret key name `DOCBASE_PASSWORD/DOCBASE_PASSWORD` in the HashiCorp Vault server.

Vault type variables for Reports

Vault type variables in the documentum/config/passwords_vault.yaml file

- `cs-secrets.clients.drServiceAccountPassword`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `cs-secrets.clients.drServiceAccountPassword`

Vault type variable in the documentum/values.yaml file

- `global.isVaultEnabled`
- `global.vaultType`

Vault type variables for OpenText Documentum CM for Microsoft 365

Vault type variables in the documentum/config/passwords_vault.yaml file

- `notificationservice.configMap.database.password`
- `notificationservice.configMap.database.dbunencryptedpassword`
- `notificationservice.configMap.database.db_ssl_rootcert`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `notificationservice.configMap.database.password`
- `notificationservice.configMap.database.dbunencryptedpassword`
- `notificationservice.configMap.database.db_ssl_rootcert`

Vault type variables for OpenText Documentum CM Online Editing Service

Vault type variables in the documentum/config/passwords_vault.yaml file

- `oesconnector.configMap.database.password`
- `oesconnector.configMap.database.dbunencryptedpassword`
- `oesconnector.configMap.database.db_ssl_rootcert`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `oesconnector.configMap.database.password`
- `oesconnector.configMap.database.dbunencryptedpassword`
- `oesconnector.configMap.database.db_ssl_rootcert`

Vault type variables for Notification Service

Vault type variables in the documentum/config/passwords_vault.yaml file

- `notificationservice.configMap.database.password`
- `notificationservice.configMap.database.dbunencryptedpassword`
- `notificationservice.configMap.database.db_ssl_rootcert`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `notificationservice.configMap.database.password`
- `notificationservice.configMap.database.dbunencryptedpassword`
- `notificationservice.configMap.database.db_ssl_rootcert`

Vault type variables for Content Connect

Vault type variables in the documentum/config/passwords_vault.yaml file

- `contentconnect.configmap.clientSecret`

- `contentconnect.configmap.DB_SSL_CA`
- `contentconnect.secret.DB_PASSWORD`
- `contentconnect.vault.VAULT_SECRET`
- `contentconnect.vault.VAULT_KEY`
- `contentconnect.certificatesecret.crt`
- `contentconnect.certificatesecret.key`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `contentconnect.configmap.clientSecret`
- `contentconnect.configmap.DB_SSL_CA`
- `contentconnect.secret.DB_PASSWORD`
- `contentconnect.vault.VAULT_SECRET`
- `contentconnect.vault.VAULT_KEY`
- `contentconnect.certificatesecret.crt`
- `contentconnect.certificatesecret.key`

Vault type variable for Documentum Connector for Core share

Vault type variables in the documentum/config/passwords_vault.yaml file

- `dctm-dcc.database.password`
- `dctm-dcc.database.dbSecretKeyName`
- `dctm-dcc.database.dbunencryptedpassword`
- `dctm-dcc.syncnshareManual.configMap.setup.sourceRepoPassword`
- `dctm-dcc.syncnshareManual.configMap.setup.targetRepoClientSecret`
- `dctm-dcc.syncnshareManual.configMap.setup.sourceRepoVaultSecretName`
- `dctm-dcc.syncnshareManual.configMap.setup.targetRepoVaultSecretName`
- `dctm-dcc.syncnshareManual.configMap.database.dbSecretKeyName`
- `dctm-dcc.mailService.configMap.mailconfig.clientSecret`
- `dctm-dcc.mailService.configMap.mailconfig.mailpassword`
- `dctm-dcc.mailService.configMap.mailconfig.mailPasswordVaultSecretName`
- `dctm-dcc.metadata.database.dbSecretKeyName`
- `dctm-dcc.coreNotification.database.dbSecretKeyName`
- `dctm-dcc.coreNotification.coresharecrt`

Vault type variables in the documentum/config/passwords_k8api.yaml file

- `dctm-dcc.database.password`
- `dctm-dcc.database.dbSecretKeyName`
- `dctm-dcc.database.dbunencryptedpassword`
- `dctm-dcc.syncnshareManual.configMap.setup.sourceRepoPassword`
- `dctm-dcc.syncnshareManual.configMap.setup.targetRepoClientSecret`
- `dctm-dcc.syncnshareManual.configMap.setup.sourceRepoVaultSecretName`
- `dctm-dcc.syncnshareManual.configMap.setup.targetRepoVaultSecretName`
- `dctm-dcc.syncnshareManual.configMap.database.dbSecretKeyName`
- `dctm-dcc.mailService.configMap.mailconfig.clientSecret`
- `dctm-dcc.mailService.configMap.mailconfig.mailpassword`
- `dctm-dcc.mailService.configMap.mailconfig.mailPasswordVaultSecretName`
- `dctm-dcc.metadata.database.dbSecretKeyName`
- `dctm-dcc.coreNotification.database.dbSecretKeyName`
- `dctm-dcc.coreNotification.coresharecrt`

Notes

- Provide the `clientSecret` if the authentication type is OAuth for mail service. Provide the encrypted client secret from the Microsoft Entra admin center after completing the app registration.
- Provide the `mailpassword` if the authentication type is basic and SMTP server requires credentials, then provide the encrypted password for the server.
- Provide the Core Share certificate if the `https` protocol is enabled.

Vault type variable in the documentum/config/configuration.yml file

- `dctm-dcc.database.databaseServiceName`

  Notes

  - This variable is applicable only if you want to use HashiCorp Vault.
  - If Documentum Connector for Core Share uses a common database with Documentum CM Server, ensure that the value of `dctm-dcc.database.databaseServiceName` is same as mentioned in Documentum CM Server.

- If Documentum Connector for Core Share is using a separate database, then the value of `dctm-dcc.database.databaseServiceName` can be different, but a new secret/key combination of `DATABASE_PASSWORD/<databaseServiceName>_<username>` must be created in the HashiCorp Vault server.

  The list of secret and key names is available in the section 2.2.2 “Storing key-value pair in Vault” in *OpenText Documentum Connector for Core Share - Deployment Guide (EDCCOCDFS-IGD)*.

Vault type variables for Documentum Archive Services for SAP Solutions

Vault type variable in the documentum/config/passwords_vault.yaml and documentum/config/passwords_k8api.yaml files

- `dctm-assap.docbasePassword`

Vault type variables for Documentum Archive Services for SAP Solutions ILM

Vault type variable in the documentum/config/passwords_vault.yaml and documentum/config/passwords_k8api.yaml files

- `dctm-assap-ilm.docbasePassword`

Vault type variables for Documentum Content Services for SAP Solutions

Vault type variable in the documentum/config/passwords_vault.yaml and documentum/config/passwords_k8api.yaml files

- `dctm-cssap.docbasePassword`

4.2 New Relic

New Relic is an application performance monitoring tool. This tool gives information that helps to analyze the application behavior, create real-time dashboards, and customization charts that are useful for application metric.

Provide the appropriate values for all the variables. To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

Global New Relic variables for all components

New Relic variables in the documentum/values.yaml file

- `global.newrelic`
- `global.newrelicProxyHost`
- `global.newrelicProxyPort`
- `global.newrelicProxyProtocol`

New Relic variables for Documentum CM Server, connection broker, and Java Method Server

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `licenseKey`
- `cs-secrets.newrelic.license_key`

New Relic variables in the documentum/config/configuration.yml file

- `docbroker.newrelic.c_app_name`
- `content-server.newrelic.app_name`
- `content-server.newrelic.c_app_name`

New Relic variables for client

New Relic variables in the documentum/config/configuration.yml file

- `d2config.newrelic.app_name`
- `d2classic.newrelic.app_name`
- `d2smartview.newrelic.app_name`
- `d2rest.newrelic.app_name`

New Relic variables for Admin Console

New Relic variable in the documentum/config/configuration.yml file

- `adminconsole.newrelic.app_name`

New Relic variables for OpenText AppWorks Gateway

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `appworks-gateway.newrelic.licenseKey`

New Relic variables in the documentum/config/configuration.yml file

- `appworks-gateway.newrelic.app_name`
- `appworks-gateway.newrelic.logFileName`
- `appworks-gateway.newrelic.logLevel`

New Relic variables in the documentum/values.yaml file

- `appworks-gateway.newrelic.enabled`

- `appworks-gateway.newrelic.proxyHost`
- `appworks-gateway.newrelic.proxyPort`
- `appworks-gateway.newrelic.proxyScheme`

New Relic variables for OpenText Intelligent Viewing

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `otiv.global.newrelic.licenseKey`

New Relic variables in the documentum/config/configuration.yml file

- `otiv.global.newRelic.baseAppName`

New Relic variables in the documentum/values.yaml file

- `otiv.global.newRelic.host`
- `otiv.global.newRelic.port`

New Relic variables for Foundation CMIS API

New Relic variables in the documentum/config/configuration.yml file

- `dctm-cmis.newrelic.app_name`
- `dctm-cmis.newrelic.configurationFile`
- `dctm-cmis.newrelic.addNodeNamePrefix`

New Relic variables for Foundation SOAP API

New Relic variable in the documentum/config/configuration.yml file

- `dfs.newrelic.dfs_application_name`

New Relic variables for Foundation REST API

New Relic variables in the documentum/config/configuration.yml file

- `dctm-rest.newrelic.app_name`
- `dctm-rest.newrelic.configurationFile`
- `dctm-rest.newrelic.addNodeNamePrefix`

New Relic variables for Documentum Administrator

New Relic variable in the documentum/config/configuration.yml file

- `da.newrelic.appName`

New Relic variables for Records Client

Records Manager

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `records.userProvidedServices.newrelic.licenseKey`

New Relic variable in the documentum/config/configuration.yml file

- `records.userProvidedServices.newrelic.appName`

Records Queue Manager

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_vault.k8api file

- `rqm.userProvidedServices.newrelic.licenseKey`

New Relic variable in the documentum/config/configuration.yml file

- `rqm.userProvidedServices.newrelic.appName`

New Relic variables in the documentum/config/configuration.yml file

- `records.userProvidedServices.newrelic.appName`
- `rqm.userProvidedServices.newrelic.appName`

New Relic variables for Documentum xPlore

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `xPlore-OneD.newRelic.licenseKey`

New Relic variables in the documentum/config/configuration.yml file

- `xPlore-OneD.newRelic.appNameSuffix`
- `xPlore-OneD.newRelic.configMap.isExist`
- `xPlore-OneD.newRelic.configMap.name`

New Relic variables for Messaging Service

New Relic variables in the documentum/config/configuration.yml file

- `dctm-dms.newrelic.configurationFile`
- `dctm-dms.newrelic.addNodeNamePrefix`

- `dctm-dms.newrelic.app_name`

New Relic variables for Reports

New Relic variable in the documentum/config/configuration.yml file

- `dtrbase.newrelic.appName`

New Relic variables in the documentum/values.yaml file

- `global.newrelic`
- `global.newrelicProxyHost`
- `global.newrelicProxyPort`
- `global.newrelicProxyProtocol`

New Relic variables for OpenText Documentum CM for Microsoft 365

New Relic variable in the documentum/config/configuration.yml file

- `smartviewm365.newrelic.app_name`

New Relic variables for OpenText Documentum CM Online Editing Service

New Relic variables in the documentum/config/configuration.yml file

- `oesconnector.newrelic.app_name`

New Relic variables for Notification Service

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `notificationservice.newrelic.license_key`

New Relic variables in the documentum/config/configuration.yml file

- `notificationservice.newrelic.app_name`
- `notificationservice.newrelic.proxy_enable`

New Relic variables for Content Connect

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `contentconnect.newrelic.new_relic_license_key`

New Relic variable in the documentum/config/configuration.yml file

- `contentconnect.newrelic.new_relic_app_name`

New Relic variables for Documentum Connector for Core Share

New Relic variable in the documentum/config/passwords.yaml or documentum/config/passwords_vault.yaml or documentum/config/passwords_k8api.yaml file

- `dctm-dcc.dcc.newrelic.license_key`

New Relic variable in the documentum/config/configuration.yml file

- `dctm-dcc.dcc.newrelic.app_name`

4.3 Graylog

Graylog is a lightweight configuration management system to collect the logs. Documentum CM Server supports the integration with Graylog using the Fluentd sidecar. As a prerequisite, add all the users to the Graylog user group. *Graylog* documentation provides detailed information.

Provide the appropriate values for all the variables. To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

Graylog variables in the documentum/values.yaml file

- `global.grayLogEnable`
- `global.graylogServer`
- `global.graylogPort`
- `global.graylogTCPPort`
- `global.businessUnit`
- `global.datacenter`
- `global.environment`
- `global.fluentd`
- `global.fluentdTcpPort`
- `global.fluentdRestPort`
- `global.fluentdUdpPort`

Graylog variable in the documentum/dockerimages-values.yaml file

- `global.fluentdImage`

4.4 Liveness probe

The liveness probe feature of Kubernetes is used to check if the container or pod, where applications are running, is running or stuck. In the case of the liveness probe failure, the container or pod is ready for a restart.

Provide the appropriate values for all the variables. To provide the appropriate values, open the YAML files, and read the descriptions of the variables.

Liveness probe of Documentum CM Server

The liveness probe of Documentum CM Server checks the `50000` and `50001` ports.

By default, the liveness probe starts to check if the Documentum CM Server container or pod is running or stuck, 10 minutes after the Documentum CM Server container or pod is deployed. However, you can modify the value of `content-server.contentserver.liveness.initialDelaySeconds` in the `documentum/config/configuration.yml` file to set a different initial delay time.

After the initial delay time that you have set, the liveness probe of Documentum CM Server is done once in three minutes. If the liveness probe fails consecutively for three times, then the container or pod is ready for a restart.

By default, the liveness probe is enabled. If you want to disable the liveness probe of Documentum CM Server, set the value of `content-server.contentserver.liveness.enable` to `false` in the `documentum/config/configuration.yml` file.

If the repository is unavailable because of the unavailability of the dependent database, then the Documentum CM Server container or pod does not get restarted automatically.

Liveness probe of connection broker

By default, the liveness probe starts to check if the connection broker container or pod is running or stuck, 10 minutes after the connection broker container or pod is deployed. However, you can modify the value of `docbroker.liveness.initialDelaySeconds` in `documentum/charts/docbroker/values.yaml` to set a different initial delay time. After the initial delay time that you have set, the liveness probe of the connection broker container or pod is done once in three minutes. If the liveness probe fails consecutively for five times, then the container or pod is ready for a restart.

By default, the liveness probe is enabled. If you want to disable the liveness probe of the connection broker container or pod, set the value of `docbroker.docbroker.liveness.enable` to `false` in the `documentum/config/configuration.yml` file. In addition, provide the appropriate value for `docbroker.fluentd_service.liveness.enable` in the `documentum/config/configuration.yml` file according to your requirement.

Table 4-1: Liveness of connection broker

Connection mode	External connection broker enabled	Default port(s) to probe
native	false	1489
secure	false	1490
dual	false	1489, 1490
native	true	1489, 1491
secure	true	1490, 1492
dual	true	1489, 1490, 1491, 1492

Liveness probe of Java Method Server

To enable liveness probe of Java Method Server, the liveness probe in Documentum CM Server container verifies the following ports and URLs:

- Documentum CM Server ports: `50000` and `50001`
- Java Method Server port: `9080`
- URL of Java Method Server: `<JMS_PROTOCOL>://localhost:9080/DmMethods/servlet/DoMethod`
- URL of OpenText Documentum CM Accelerated Content Services: `<JMS_PROTOCOL>://localhost:9080/ACS/servlet/ACS`

By default, the liveness probe is enabled. If you want to disable the liveness probe of Java Method Server, set the value of `content-server.contentserver.liveness.considerJMSForLiveness` to `false` in the `documentum/config/configuration.yml` file.

Liveness probe of Foundation CMIS API

The liveness probe of Foundation CMIS API sends the HTTP/HTTPS requests to the Foundation CMIS API pod to check if the Foundation CMIS API container or pod is running or stuck. By default, the liveness probe is enabled.

Liveness probe variables in the documentum/config/configuration.yml file

- `dctm-cmis.livenessProbe.enabled`
- `dctm-cmis.livenessProbe.scheme`
- `dctm-cmis.livenessProbe.initialDelaySeconds`
- `dctm-cmis.livenessProbe.periodSeconds`

The port number in which the requests are sent to, is the same as defined in `dctm-cmis.httpPort` or `dctm-cmis.httpsPort` in the `documentum/values.yaml` file depending on scheme utilized.

Liveness probe of Foundation SOAP API

The liveness probe of Foundation SOAP API sends the HTTP/HTTPS requests to the Foundation SOAP API pod to check if the Foundation SOAP API container or pod is running or stuck. By default, the liveness probe is enabled.

Liveness probe variables in the documentum/config/configuration.yml file

- `dfs.containers.dfs.probing.livenessProbe.initialDelaySeconds`
- `dfs.containers.dfs.probing.livenessProbe.periodSeconds`

Liveness probe of Foundation REST API

The liveness probe of Foundation REST API sends the HTTP/HTTPS requests to the REST pod to check if the Foundation REST API container or pod is running or stuck. By default, the liveness probe is enabled.

Liveness probe variables in the documentum/config/configuration.yml file

- `dctm-rest.livenessProbe.enabled`
- `dctm-rest.livenessProbe.scheme`
- `dctm-rest.livenessProbe.initialDelaySeconds`
- `dctm-rest.livenessProbe.periodSeconds`

The port number in which the requests are sent to, is the same as defined in `httpPort` or `httpsPort` in the `dctm-rest` section in the `documentum/values.yaml` file depending on scheme utilized.

Liveness probe of Documentum Administrator

The liveness probe of Documentum Administrator sends the HTTP/HTTPS requests to the Documentum Administrator pod to check if the Documentum Administrator container or pod is running or stuck. By default, the liveness probe is enabled.

Liveness probe variables in the documentum/config/configuration.yml file

- `da.containers.da.probing.livenessProbe.initialDelaySeconds`
- `da.containers.da.probing.livenessProbe.periodSeconds`

Liveness probe of Records Client

The liveness probe of Records Client sends the HTTP/HTTPS requests to the Records Client pod to check if the Records Client container or pod is running or stuck. By default, the liveness probe is enabled.

Liveness probe variables in the documentum/values.yaml file

- `records.containers.records.probing.livenessProbe.initialDelaySeconds`
- `records.containers.records.probing.livenessProbe.periodSeconds`

Liveness probe of Messaging Service

The liveness probe of Messaging Service sends the HTTP/HTTPS requests to the Messaging Service pod to check if the Messaging Service container or pod is running or stuck. By default, the liveness probe is enabled.

Liveness probe variables in the documentum/config/configuration.yml file

- `dctm-dms.dms.probing.livenessProbe.enable`
- `dctm-dms.dms.probing.livenessProbe.initialDelaySeconds`
- `dctm-dms.dms.probing.livenessProbe.periodSeconds`

Liveness probe of Independent Java Method Server

The liveness probe of Independent Java Method Server sends the HTTP/HTTPS requests to the Independent Java Method Server pod to check if the Independent Java Method Server container or pod is running or stuck. By default, the liveness probe is enabled.

Liveness probe variables in the documentum/config/configuration.yml file

- `ijms.liveness.enable`
- `ijms.liveness.initialDelaySeconds`
- `ijms.liveness.periodSeconds`
- `ijms.liveness.failureThreshold`

Liveness probe of OpenText Content Aviator

Liveness probe variables in the documentum/addons/aviator/aviator-config.yaml file

- `dctm-dcis.livenessProbe.enabled`
- `dctm-dcis.livenessProbe.scheme`
- `dctm-dcis.livenessProbe.probeUrl`
- `dctm-dcis.livenessProbe.initialDelaySeconds`
- `dctm-dcis.livenessProbe.periodSeconds`

4.5 Event Hub

OpenText Documentum CM supports the Event Hub feature using the Fluentd and Apache Kafka services.

You must configure the Apache Kafka services externally. You must configure the Salted Challenge Response Authentication Mechanism (SCRAM) plain authentication mechanism in your Apache Kafka deployment. In addition, you must create a user and a topic for producing the messages which is pushed to Kafka.

Fluentd is an open source data collection tool using the Apache 2.0 license. Fluentd acts as an event aggregator for the OpenText Documentum CM applications and producer for Kafka using `@forward` and `@Kafka2` as input and output plug-ins respectively. Kafka is used to store the events accumulated at Fluentd.

Audit trail, custom, and remote procedure call (RPC) events are logged in Kafka.

The following fields are mandatory for the events logged in Event Hub:

- `EVENT_VERSION`
- `EVENT_DESCRIPTION`
- `EVENT_DATETIME`
- `EVENT_NAME`
- `EVENT_CATEGORY`

Integrating Event Hub with Documentum CM Server

- To use the feature, you must set the value of `global.fluentd` to `true` in "Global variables" on page 49 in the `documentum/values.yaml` file. Make sure that you update the values of other Fluentd and Kafka related variables in "Global variables" on page 49 in the `documentum/values.yaml` file.

  Fluentd pushes the events that are generated by Documentum CM Server to the external Apache Kafka cluster.

  By default, the `Eventhub.dar` file is deployed when you deploy the Documentum CM Server pod. The `Eventhub.dar` file establishes the connection with Fluentd and works as a BOF module.

  The `dfc.properties` file of Documentum CM Server contains the following variables for Event Hub:

  – `dfc.client.should_use_eventhub`: Used to enable Event Hub at Foundation Java API. Set the value to `true` in all the pods.

  – `dfc.client.eventhub.log_level`: Used to define the log level for Foundation Java API events. This variable uses the value you provided for `eventLogLevel` in "Global variables" on page 49. The values may range from `0` to `5` where `0` is for NO LOG, `1` is for ERROR, `2` is for WARN, `3` is for INFO, `4` is for DEBUG, and `5` is for TRACE.

- Fluentd is packaged with Documentum CM Server.

  Make sure that you provide the appropriate values for all the Fluentd variables. Some of the important variables are `cs-logging-configMap.fluentdConf.bufferingMode` and `cs-logging-configMap.fluentdConf.compressionMode`.

- You can retrieve the messages in the following ways:

  – Using the Kafka built-in utility, sign in to the external Kafka deployment, go to `/kafka-setup/kafka_<version>/bin`, and then run the following command:

  ```
  ./kafka-console-consumer.sh --topic <topic name> --from-beginning --
  consumer.config ../config/consumer.properties --bootstrap-server <kafka
  broker IP address or host>:<kafka broker port>
  ```

  – Using the sample program: To access events from external Kafka deployment using the sample program, see *Apache Kafka* documentation.

  Note: You may encounter any additional messages that are not related to custom, Foundation Java API, or Audit events while retrieving the logged event messages. You can ignore these additional messages.

Integrating Event Hub with Foundation SOAP API

- To use the feature, set the value of `dfc.client.should_use_eventhub` to `true` in `dfc.properties` of Foundation SOAP API.

  Fluentd pushes the events that are generated by Foundation SOAP API to the external Apache Kafka cluster.

  By default, the `Eventhub.dar` file is deployed when you deploy the Documentum CM Server pod. The `Eventhub.dar` file establishes the connection with Fluentd and works as a BOF module.

  The `dfc.properties` of Foundation SOAP API and the `documentum/values.yaml` files contain the following variables for Event Hub:

  – `dfc.client.should_use_eventhub`: Used to enable Event Hub. Set the value to `true` to enable Event Hub. The default value is `false`.

  – `dfc.client.eventhub.log_level`: Used to define the event log levels. Set any value from `0` to `5` where `0` is for NO LOG, `1` is for ERROR, `2` is for WARN, `3` is for INFO, `4` is for DEBUG, and `5` is for TRACE.

  – `dfc.client.eventhub.queue_size`: Used to define the number of events in the queue buffered in Foundation Java API.

  Note: Changes to the `dfs-configmap.yaml` file are reflected in the Foundation SOAP API pod within a few seconds.

- Fluentd is packaged with Foundation SOAP API.

  Make sure that you provide the appropriate values for all the Fluentd and Kafka variables.

- The following table lists the Foundation SOAP API specific event names:

Service name or category	Event name
REPOSITORY INQUIRY	GetRepoListOperation
	ListContentReposAction
	SetContentRepoList
	GetServers
	CheckMinServerVersionCompatibility
	GetRepoNameByObjIdOperation
	RepoNameById
QUERY	ExecuteOperation
	QueryAction
	GetActionResult
	LogQuery
	SetQueryResult
QUERY STORE	ListSavedQueriesOperation
	ListSavedQueriesAction
	ListSavedSearches
	LoadSavedQueryOperation
	LoadSavedQueryAction
	LoadSavedQuery
SCHEMA	GetRepoInfoOperation
	GetSchemaInfoOperation
	GetTypeInfosOperation
	GetTypeInfoOperation
	GetPropertyInfoOperation
	GetDynamicAssistValuesOperation
	GetValueAssistSnapshotOperation
OBJECT CREATE	CreateOperation
	CreateAction
	CreateObject
	Validate
	ValidateContent
	CreateContentfulObject

Service name or category	Event name
	CreateContentlessObject
	UpdateLightweigthObject
	Checkout
	SaveObject
OBJECT CREATE PATH	CreatePathOperation
OBEJCT SERVICE GET	GetOperation
	DeepGetAction
	ContentLoaderObjectCreate
	PropertiesLoaderObejctCreate
	PermissionLoaderObjectCreate
	Walk
	AddCreatedObject
	GatherRelatedDataObjects
	LoadRestOfRelations
OBJECT MOVE	MoveOperation
	MoveAction
	MoveProcess
OBJECT UPDATE	UpdateOperation
	UpdateAction
	UpdateObject
OBJECT COPY	CopyOperation
	CopyAction
	CopyProcess
OBJECT SAVE AS NEW	SaveAsNewOperation
	SaveAsNewAction
OBEJCT SERVICE REGISTER EVENT	RegisterEventOperation
	RegisterEventAction
OBJECT DELETE	DeleteOperation
	DeleteAction
	DeleteSysObject
OBJECT UNREGISTER	UnRegisterEventOperation
	UnRegisterEventAction
OBJECT VALIDATE	ValidateOperation

Service name or category	Event name
	ValidateAction
OBJECT GET PERMISSION	GetPermissionOperation
	GetPermissionAction
OBJECT GET LINKED REPEATING STRING	GetLinkedRepeatingStringOperation
	GetLinkedRepeatingStringAction
OBJECT GET CONTENT URLS	GetContentUrlsOperation
	GetContentUrlsProcess
OBJECT MARK VERSION	MarkVersionOperation
	MarkVersionAction
OBJECT UPDATE NON CURRENT VERSION	UpdateNonCurrentVerionOperation
	UpdateNonCurrentVerionAction
OBJECT HAS ATTRIBUTES	HasAttributesOperation
	HasAttributesAction
OBJECT REMOVE RENDITION	RemoveRenditionOperation
	RemoveRenditionAction
USER AUTHENTICATION	UserAuthentication
RELATE ACTION	RelateAction
	ProcessLighObjRel
GENERIC	GetSession
	ReleaseSession
	NewSessionCreate
	FindDataObjectById
	GetChronicalId
	GetCurrentVersion
	GetCurrentVersionIdentity
	CurrentVersionCheck
	ImmutableCheck
	GetPersistentObj
	GetRawId
	GetIdByQualification
	GetMultiIdsByQualification
	GetTargetObjIdentity
DOCUMENTUM UTIL	DateTimeToIDfTimeOperation

Service name or category	Event name
	DateTimeToIDfTimeAction
LOGIN TICKET	GetLoginTicketOperation
	GetDCTMLoginTicketOperation
	GetLoginTicketAction
	GetDCTMLoginTicketAction
SEARCH	AsyncSearchAction
	StopSearchOperation
	StopSearchAction
	ListSearchSourcesAction
	GetClustersOperation
	GetClusterAction
	GetSubclustersOperation
	GetResultsPropsOperation
	GetFacetsOperation
	GetFacetsAction
COMMENT	CreateCommentOperation
	EnumCommentOperation
	GetCommentOperation
	MarkReadOperation
	MarkUnreadOperation
	UpdateCommentOperation
TASK MANAGEMENT	ClaimOperation
	WorkQueueTaskAcquire
	WorkflowTaskAcquire
	ReleaseOperation
	SuspendOperation
	SuspendUntilOperation
	WorkQueueTaskSuspend
	ResumeOperation
	WorkflowResume
	CompleteOperation
	WorkflowTaskComplete
	SetNextActivities

Service name or category	Event name
	SetRerunMethod
	RemoveOperation
	SetPriorityOperation
	AddAttachmentOperation
	GetAttachmentInfosOperation
	GetAttachmentsOperation
	DeleteAttachmentsOperation
	RemoveWorkflowAttachment
	AddCommentOperation
	AddComment
	GetCommentsOperation
	ForwardOperation
	DelegateOperation
	SetDelegateUser
	GetTaskInfoOperation
	GetTaskDescriptionOperation
	SetOutputOperation
	GetInputOperation
	GetOutputOperation
	NominateOperation
	GetMyTaskAbstractsOperation
	GetMyTasksOperation
	QueryOperation
WORKFLOW	StartProcessOperation
	StartedWorkflowService
	SetPackageInfo
	SetAttachmentInfo
	SetAliasInfo
	SetPerformerInfo
	GetProcessInfoOperation
	RetrievePackageInfo
	RetrieveAliasInfo
	RetrievePerformerInfo

Service name or category	Event name
	GetProcessTempsOperation
	GetProcessDQL
	TerminateWorkflowOperation
	WorkflowHalt
	WorkflowAbort
	WorkflowDestroy
	CompleteWorkItemOperation
	WorkItemAcquire
	GetForwardActivities
	SetOutputActivities
	WorkItemComplete
LIFECYCLE	AttachOperation
	AttachAction
	ExecuteLifecycleAction
	GetLifecycleOperation
	GetLifecycleAction
	DetachOperation
	DetachAction
VIRTUAL DOCUMENT	UpdateDocumentOperation
	RetrieveSnapshotOperation
	CreateSnapshotOperation
	CreateSnapshotAction
	RemoveSnapshotOperation
	DisassemblyAction
	PrepareVDocUpdateAction
USER MANAGEMENT	GetUserOperation
	GetUserAction
VIRTUAL DEPLOYMENT	GetDormacyStatusOperation
	GetDormacyStatusAction
	MakeActiveOperation
	MakeDormantOperation
	ProjectDormant
	ChangeDormancyStatusAction

Service name or category	Event name
	Events Related to Agent Service Call
	GetHttpSessionIdOperation
	GetSessionIdFromReqOperation
	GetStatusOperation
	GetDeploymentIdOperation
	LookupOperation
CONTEXT REGISTRY	RegisterOperation
	UnRegisterOperation
	Events Related to Analytic Service Call
	AnalyzeOperation
	Events Related to License Service Call
	RequestLicenseOperation
	CheckinOperation
VERSION CONTROL	CheckinAction
	CheckoutOperation
	CheckoutAction
	CancelCheckoutOperation
	CancelCheckoutAction
	GetCheckOutInfoOperation
	GetCheckoutInfoAction
	GetVersionInfoOperation
	GetVersionInfoAction
	DeleteVersionOperation
	DeleteAllVersionsOperation
	GetCurrentOperation
	ACLCreateOperation
ACCESS CONTROL	CreateACLAction
	ACLUpdateOperation
	UpdateACLAction
	ACLGetOperation
	GetACLAction
	ACLDeleteOperation
	DeleteACLAction

Integrating Event Hub with Foundation REST API

- To use the feature, set the value of `dfc.client.should_use_eventhub` to `true` in `dfc.properties` of Foundation REST API.

  Fluentd pushes the events that are generated by Foundation REST API to the external Apache Kafka cluster.

  By default, the `Eventhub.dar` file is deployed when you deploy the Documentum CM Server pod. The `Eventhub.dar` file establishes the connection with Fluentd and works as a BOF module.

  The `dfc.properties` of Foundation REST API file contains the following variables for Event Hub:

  – `dfc.client.should_use_eventhub`: Used to enable Event Hub. Set the value to `true` to enable Event Hub. The default value is `false`.

  – `dfc.client.eventhub.log_level`: Used to define the log levels for the Foundation Java API events. Set any value from `0` to `5` where `0` is for NO LOG, `1` is for ERROR, `2` is for WARN, `3` is for INFO, `4` is for DEBUG, and `5` is for TRACE.

  – `dfc.client.eventhub.queue_size`: Used to define the number of events in the queue buffered in Foundation Java API.

- Fluentd is packaged with Foundation REST API.

  Make sure that you provide the appropriate values for all the Fluentd and Kafka variables.

- The following table lists the Foundation REST API specific event names:

Service name or category	Event name
REST_URI_ACCESS	REQUEST_BEGIN
	REQUEST_END
REST_BATCH_URI_ACCESS	REQUEST_BEGIN
	REQUEST_END
REST_LOGIN	LOGIN_SUCCESS
	LOGIN_FAILED

- REST URI access: For Foundation REST API environments enabled with Event Hub captures all the API requests including request and response and then publishes them to Event Hub.

  By default, Foundation REST API captures the following information for a request:

  – `REQUEST_PROTOCOL`: Protocol of the request.

    For example, HTTP or HTTPS.

  – `REQUEST_URI`: Relative path of URI.

For example: `/dctm-rest/repositories/testenv`

- `REQUEST_PARAMS`: Parameter to the API endpoints.
- `REQUEST_BODY`: Payload data of the request.
- `REQUEST_METHOD`: Type of the request.

  For example, GET, POST, and so on.
- `REQUEST_HEADERS`: Request headers in the raw format.

By default, Foundation REST API captures the following information for a response:

- `RESPONSE_STATUS_CODE`: Status code of the request.
- `RESPONSE_BODY`: Response body of the request.

Foundation REST API adds a unique ID as `REQUEST_UNIQUE_ID` for all the requests to track both the request and response.

The `REQUEST_BODY` and `RESPONSE_BODY` information are captured depending on the value of `rest.eventhub.httpbody.enabled` in `rest-api.runtime.properties`. By default, the value of `rest.eventhub.httpbody.enabled` is `false` and so the `REQUEST_BODY` and `RESPONSE_BODY` information are not captured.

By default, the size of the request body and response body capture is limited to 102400 bytes (which means 100 KB). This can be configured in `rest.eventhub.httpbody.logging.limit` in `rest-api.runtime.properties`.

- REST batch URI access: The REST URI Access feature is also implemented in the Batch request API. For all the batch requests, Foundation REST API publishes individual API requests and responses.
- Authentication: Foundation REST API sends all the authentication events to Event Hub. For Event Hub, Foundation REST API support the authentication mode as described in the following table:

Authentication type mentioned in rest-api.runtime.properties	AUTHENTICATION_TYPE in Event Hub events
`basic/basic-ct`	PASSWORD
`otds_password/ct-otds_password`	OTDS_PASSWORD
`otds_token/ct-otds_token/otds_token-basic/ct-otds_token-basic`	OTDS_TOKEN

The event category for authentication events is `EVENT_CATEGORY` and the category for event names are `LOGIN_SUCCESS` and `LOGIN_FAILED`. If the authentication fails for OTDS token-based login, then the `OTDS_TOKEN` field records the login-based information. All the authentication events are synchronous.

- Publishing event from external client: From the 22.1 release, an external client can publish an event to Event Hub using Foundation REST API endpoint. The following API is used to call with the POST data:

  ```
  End Point: /repositories/<REPOSITORY_NAME>/event-hub-event
  ```

  For example:

  ```
  {
      "event-category": "<CLIENT_EVENT_CATEGORY>", //MANDATORY FIELD
      "event-name": "<CLIENT_EVENT_NAME>", //MANDATORY FIELD
      "event-level": "2", //MANDATORY FIELD
      "event-data": "<CLIENT_EVENT_DATA>", //MANDATORY FIELD
      "username": "<username@org.com>",
        "object-id": "",
      "duration": "22",
      "application-name": "X Client",
      "client-ip": "<IP address of client>",
      "client-location": "",
      "synchronous": "true",
      "event-fields": {
          "EXTRA_EVENT_1": "Sample1",
          "EXTRA_EVENT_2": "Sample2"
      }
  }
  ```

  The `synchronous` field indicates if Event Hub is sent synchronously (`true`) or asynchronously (`false`). The clients can customize the fields in the `event-fields` field. The API returns the `201 CREATED` response.

4.6 Scaling and descaling

You can increase (scale) and decrease (descale) the number of replica pods to be spawned for the Documentum CM Server image.

Note: The scaling and descaling features are not supported for the connection broker.

4.6.1 Scaling Documentum CM Server

1. To increase the number of replica pods to be spawned for the Documentum CM Server image, update `content-server.contentserver.replicaCount in <location where Helm charts are extracted>/documentum-resources-values-<config>.yaml` to a required value.

2. After increasing the value of `replicaCount`, run the Helm upgrade command.

4.6.2 Descaling Documentum CM Server

1. To decrease the number of replica pods to be spawned for the Documentum CM Server image, update `content-server.contentserver.replicaCount in <location where Helm charts are extracted>/documentum-resources-values-<config>.yaml` to a required value.

2. After decreasing the value of `replicaCount`, run the Helm upgrade command.

Note: By default, the resources are not removed after the descaling process. OpenText does not recommend you to remove the resources.

4.7 Multiple repositories

4.7.1 Multiple repository support in Documentum CM Server

One statefulset of Documentum CM Server node serves only one repository. To deploy multiple repositories, you must deploy new statefulset in a separate namespace.

The global repository and the new repository must be projected to the same connection broker. When you are deploying a new repository, it can either use the current repository as the global repository or another repository as the global repository. If you want the new repository to use another repository as the global repository, then you must configure the values for all the variables in `content-server.globalRepository` in the `documentum/config/configuration.yml` file.

Update the value of the primary connection broker’s cluster domain for `content-server.docbroker.clusterSpace` in the `documentum/values.yaml` file for the member repository.

The new repository can either use the database of the previous repository or a separate database. If you want the new repository to use the database of the previous repository, then you must provide unique values for the repository name and repository owner.

In the `documentum/config/configuration.yml` file, provide unique values for the following variables:

- `content-server.docbase.id`
- `content-server.docbase.index`

In addition to the information in this section, for the multiple repository support in Documentum CM Server, use the information and instructions as described in “Multiple repository support in client” on page 190.

4.7.2 Multiple repository support in client

4.7.2.1 Overview of high-level steps to support multiple repository for cloud deployments

For the multiple repositories setup, the deployment steps are as follows:

1. Deploy the Helm chart with required clients and other components on your chosen hyperscaler. The repository configured in this deployment is considered as a global repository and the deployment is considered as a global repository deployment.

2. For each additional repository, deploy the Helm with only Documentum CM Server and related components (`cs-secrets`, `cs-logging-configMap`, `content-server`, `cs-dfc-properties`, `dctm-ingress`, `xda`, `bps`, and `xplore`) enabled. Make sure that all additional repositories are configured to the single connection broker that was previously deployed.

And as a best practice, deploy additional repositories in separate namespaces in the same cluster.

> **Important**
> All the following sections, including the prerequisites, have to be repeated for each additional repository deployment.

4.7.2.2 Prerequisites

1. Make sure that the single Helm deployment with all required clients and components are enabled on the chosen hyperscaler. Make sure that all replicas of Documentum CM Server and the connection broker pods are in running state. The repository configured in this deployment is considered as global a repository and the deployment is considered as a global repository deployment.

2. The global repository deployed cluster will be considered for all additional repository deployments.

3. The install owner user name and password should be the same as the global repository for all the additional repositories. This should be the same for all repositories.

   ```
   installOwnerUsername: &installOwner_username dmadmin
   installOwnerPassword: &installOwner_password password
   ```

4. Create a new namespace for each additional repository deployment in the same cluster (where the global repository is deployed).

5. Copy the AEK key (`aek_name`) from the Documentum CM Server pod (`dcs-pg`) of the global repository from the location `/opt/dctm/dba/secure/aek_name`. This can be done by replacing the `<global repository namespace>` with the namespace of the global repository deployment and the `<aek.name provided in the global repository deployment>` with the `content-server.contentserver.aek.name` provided in the global repository deployment in the following example command:

   ```
   kubectl cp <global repository namespace>/dcs-pg-0:opt/dctm/dba/secure/<aek.name
   provided in global repository deployment> aek_name -c dcs-pg
   ```

   After updating the values, run the command.

6. Create a PVC in the new namespace where the additional repository will be deployed using the following steps:

   a. Create a YAML file with the following details. Replace `<serviceName of the Documentum CM Server pod in the global repository deployment>` with the `serviceName` of the Documentum CM Server pod in the global repository deployment:

   ```
   apiVersion: v1
   kind: PersistentVolumeClaim
   metadata:
     name: <serviceName of the Documentum CM Server pod in global repository
   deployment>-sharedkey-pvc
   spec:
     accessModes:
     - ReadWriteMany
     resources:
       requests:
        storage: 20Mi
     storageClassName: trident-nfs
     volumeMode: Filesystem
   status:
     accessModes:
     - ReadWriteMany
     capacity:
       storage: 20Mi
     phase: Bound
   ```

   b. Run the following command to create the PVC in the new namespace. Replace `<additional repository namespace>` with the additional repository namespace.

   ```
   kubectl apply -f .\pvc.yaml -n <additional repository namespace>
   ```

7. To copy the AEK copied from the global repository deployment to the newly created PVC, do the following:

   a. Create a Docker file with the following details and update `<install owner>` with the install owner of the global repository deployment:

   ```
   FROM <alpine image>
           ARG AEKPATH
           ENV AEK_KEY_PATH /aekkeypath
           RUN set -ex && apk --no-cache add sudo
           RUN adduser -D -H <install owner> &&\
               mkdir -p $AEK_KEY_PATH && \
               chown -R <install owner>:<install owner> $AEK_KEY_PATH
           COPY --chown=<install owner>:<install owner> $AEKPATH $AEK_KEY_PATH
           CMD sh
   ```

b. Create a container image using the Docker file created in the previous step. Run the following command:

```
docker build -f <path to docker file created in previous step> -t <docker
image name> --build-arg AEKPATH=<relative path to folder that contains aek key
copied from global repo> --no-cache .
```

Example:

```
docker build -f Dockerfile -t artifactory.otxlab.net/ot2-paas-dev/copyaekkey:
23.4 --build-arg AEKPATH=./folder --no-cache .
```

c. Upload the Docker image you created to the preferred registry using the following command:

```
docker push <docker image name>
```

8. The connection broker `replicaCount` mentioned in the `content-server` section of the additional repository deployment's resource YAML file should be the same as that of the connection broker replica count mentioned in the `content-server` section of the global repository deployment's resource YAML file.

   Chosen resource YAML file:

```
content-server:
  contentserver:
    replicaCount: 1
    docbrokersCount: 1
```

4.7.2.3 Additional prerequisites for deploying an additional repository on Amazon Web Services cloud platform

- If you are using static provisioning, create two access points with default values in the EFS for the following PVCs. You can ignore this step in the case of dynamic provisioning. *Amazon Web Services* documentation contains detailed information.

  Note: This EFS can be the same as the EFS used/created for global repository deployment.

  PVC for the Documentum CM Server in the EFS for xPlore.

4.7.2.4 Additional prerequisites for deploying additional repository on Azure

Configure additional ngnix-ingress so the applications are accessible externally.

4.7.2.4.1 Create a public IP address on Azure

As a best practice, create a static IP address.

Run the following command to create a static public IP address on Azure:

```
az network public-ip create --resource-group <resource group> --name <name of public IP>
--sku Standard --allocation-method static --query publicIp.ipAddress -o tsv
```

Use the following command to deploy the additional Ingress Controller:

Important
Ensure that the ingress class name used for creating the additional Ingress Controller is unique in relation to the existing ingress class names.

```
helm install <name of ingress resource> ingress-nginx/ingress-nginx --namespace
<namespace>
--set controller.replicaCount=1
--set controller.ingressClass=<ingress class name>
--set controller.ingressClassResource.name=<ingress class name>
--set controller.ingressClassResource.controllerValue="k8s.io/<ingress class name>"
--set controller.ingressClassResource.enabled=true
--set controller.IngressClassByName=true
--set controller.service.loadBalancerIP="<public Ip created in previous step>"
--set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-
health-probe-request-path"=/healthz
```

Verify that the ngnix-ingress has been deployed successfully and the service has been created.

Configure a fully-qualified domain name (FQDN) for the Ingress Controller IP address. You can run the following Bash script to map the external IP to the DNS name or run the command line-by-line in the Windows environment.

Note: Provide the IP and the DNS name details. You can find the public IP address from `kubectl get svc` when the Helm install ingress command is run. DNSNAME should be unique in relation to the existing DNS names.

```
#!/bin/bash
# Public IP address of your additional Ingress Controller created in previous step
IP="52.188.221.96"
# Name to associate with public IP address
DNSNAME="documentumdeployment"
# Get the resource-id of the public ip
PUBLICIPID=$(az network public-ip list --query "[?ipAddress!=null]|[?
contains(ipAddress, '$IP')].[id]" --output tsv)
# Update public ip address with DNS name
az network public-ip update --ids $PUBLICIPID --dns-name $DNSNAME
```

4.7.2.5 Updates to be done in Helm charts

4.7.2.5.1 documentum/documentum-components.yaml file changes

- Disable all components except `cs-secrets`, `cs-logging-configMap`, `content-server`, `cs-dfc-properties`, `dctm-ingress`, `xda`, `bps`, and `xplore`.

4.7.2.5.2 documentum/values.yaml file changes to support additional repositories

General updates

1. Replace all instances of `<namespace>` with the namespace where the additional deployment is going to be deployed.
2. Set `db_hostname` to `db-pg-0.db-pg.<namespace of global repo>.svc.cluster.local`.

Updates to anchor tag section

1. Update `rwo/rwmStorage` class as per the storage class being used.
2. Update install owner user name and install owner password with install owner user name and install owner password of the global repository deployment.
3. Update `ingressUrl: &ingressUrl` with the FQDN of the ingress.
4. Update `ingressDomain: &ingress_domain` with the ingress domain name used in the deployment.
5. Update `otdsAuthSvc: &otds_auth_svc` with the FQDN of `ingress>/otdsws`.
6. Update `globalRegistryPassword: &global_registry_password` with the global registry password.
7. Update `otdsEnabled: &otds_enabled` to true if OTDS is enabled in the global repository deployment.
8. To enable SSL mode in the additional deployment, update `useCertificate: &use_certificate` to true in the anchor tag section.

Note: Both the global repository deployment and additional repository deployments should be SSL-enabled or disabled as mixed mode is not supported.

For example:

```
rwoStorage: &rwo_storage_class azurefile
rwmStorage: &rwm_storage_class azurefile
ingressUrl: &ingressUrl namespace2.australiaeast.cloudapp.azure.com/
otdsAuthSvc: &otds_auth_svc namespace2.australiaeast.cloudapp.azure.com/otdsws
ingressDomain: &ingress_domain australiaeast.cloudapp.azure.com
useCertificate: &use_certificate false
globalRegistryPassword: &global_registry_password Password@1234567890
otdsEnabled: &otds_enabled true
```

Updates to cs-secrets section

1. The secrets file contains sensitive information like license keys and passwords. Perform the following steps to configure the Documentum CM Server in SSL mode. In the connection broker certificate section, provide the `password`, `aekpassphrase` and `trustpassword` mentioned in the global repository deployment. Provide valid certificate key and certificate in the PEM format for `pemCertPrivKey` and `pemCertificate` parameters. `pemCertPrivKey` and `pemCertificate` parameters can be left blank for the self-signed key.

```
cs-secrets:
   docbroker:
      certificate:
        password:
        aekpassphrase:
        trustpassword:
        pemCertPrivKey:
        pemCertificate:
```

2. Similarly in the `content-server.certificate` section, update the password and trust password and provide the certificate and certificate key in PEM format. `pemCertPrivKey` and `pemCertificate` parameters can be left blank for the self-signed key.

```
cs-secrets:
    contentserver:
      certificate:
        password:
        trustpassword:
        pemCertPrivKey:
        pemCertificate:
```

3. Update the database section with the database user name and password and certificate.

   Note: The database for the additional repository deployment doesn't have to be the same as the global repository deployment.

   For example:

   Database section in `cs-secrets`:

```
cs-secrets:
    database:
        userName: database-username
        password: database-password
        certificate: |
             -----BEGIN CERTIFICATE-----
             database-certificate
             -----END CERTIFICATE-----
```

   Note: If the provided Google Cloud Platform certificate is valid, the storage bucket with the given name will be created as part of the Documentum CM Server scripts. There’s no need to manually create the bucket in the Google cloud store.

4. If TLS for dctm-ingress must be enabled for additional repository deployment, provide the base64-encoded TLS certificate and the TLS key in the `cs-secrets` section as follows:

```
cs-secrets:
  ingress:
    tlscrt: <base64-encoded TLS certificate> as one line i.e. without any carriage
returns or wrapped lines.
    tlskey: <base64-encoded TLS key> as one line i.e. without any carriage returns
or wrapped lines.
```

Updates to content-server parameters

1. Update the connection broker cluster space in the `content-server` section to point to the global repository deployment's connection broker cluster space, as shown in the following example:

   Global repository values:

```
env: &env_domain namespace1.svc.cluster.local
   content-server:
      enabled: true
      docbroker:
         serviceName: *dbr_service_name
          clusterSpace: *env_domain
```

   Additional repository values:

```
env: &env_domain namespace2.svc.cluster.local
     content-server:
        enabled: true
        docbroker:
          serviceName: *dbr_service_name
          clusterSpace: namespace1.svc.cluster.local #env_domain of existing
deployment
```

2. Update `docbase` parameters in the `content-server` section, as shown in the following example code:

   Note: The `id` used for the additional `docbase` should be unique in relation to the existing repositories.

   Global repository values:

```
content-server:
    enabled: true
    docbase:
      id: 123456
      index: DM_docbase1_DOCBASE
```

   Additional repository values:

```
content-server:
    enabled: true
    docbase:
      id: 123457
      index: DM_docbase2_DOCBASE
```

3. Update the `database` parameters in the `content-server` section to point to the preferred database that includes updates to `host`, `databaseServiceName`, `port`, `sslEnabled`, `paasEnv`. and `docbaseOwnerPasswordChange`. In the illustrated example, the database of the additional repository deployment is pointing to the database of the global repository deployment.

   Global repository values:

```
content-server:
    ### Database ###
    database:
      host: db-pg-0.db-pg.namespace1.svc.cluster.local
      databaseServiceName: MyPostgres
      port: 5432
      sslEnabled: false
      paasEnv: false
      docbaseOwnerPasswordChange: false
```

Additional repository values:

```
content-server:
    ### Database ###
    database:
      host: db-pg-0.db-pg.namespace1.svc.cluster.local
      databaseServiceName: MyPostgres
      port: 5432
      sslEnabled: false
      paasEnv: false
      docbaseOwnerPasswordChange: false
```

4. Update the `globalRepositoryName` in the `content-server` section with the global repository deployment's repository name.

   Note: Make sure the global repository mentioned is up and running during the additional repository deployment.

   Global repository values:

```
content-server:
    globalRepository:
      globalRepositoryName: ""
```

   Additional repository values:

```
content-server:
     globalRepository:
      globalRepositoryName: docbase1
```

5. Set `disableUpdateAcsUrl` to true in the `ingress` subsection of the `content-server` section, and update the `ingress:host` as per your environment.

```
content-server:
   ingress:
    host: dctmint-ingress.d2.cfcr-lab.bp-paas.otxlab.net
    disableUpdateAcsUrl: false
```

   Note: The preceding parameter is set to true to avoid overwriting of `acs_base_url` when the pod restarts so that the user can manually update the required `acs_base_url` and preserve it.

6. Update the `persistentVolume.shareKeyPVCName` in the `content-server` section to point to `<serviceName of the Documentum CM Server pod in global repository deployment>-sharedkey-pvc`.

   Global repository values:

```
content-server:
    persistentVolume:
       shareKeyPVCName: ""
```

   Additional repository values:

```
content-server:
    persistentVolume:
      shareKeyPVCName: "dcs-pg-sharedkey-pvc"
```

7. Make sure the `extraEnv` section has the following values set to `T`. Also update the `<docbase_name>` placeholder to the repository name.

```
extraEnv:
  - name: LSS_CC_ENABLED
    value: "T"
  - name: DA_PRIVILEGE_ENABLED
    value: "T"
  - name: <docbase_name>_resource_id
    value: ""
  - name: <docbase_name>_secretKey
    value: ""
  - name: <docbase_name>_MIGRATE_LDAP_CONFIGS
    value: ""
  - name: MIGRATE_LDAP_DOCBASES
    value: "<docbase_name>"
```

8. Update `otds.otdsAPIsvc` in the `content-server` section to point to `<service name of OTDS in global repository deployment>.<namespace of global repository namespace>:<otds service port>/otdsws` and update `cert_jwks_url` in the `content-server` section to point to `http://<service name of otds in global repository deployment>.<namespace of global repository namespace>:<otds service port>/otdsws/oauth2/jwks`.

   Note: The service name of OTDS in the global repository deployment can be found in `documentum/values.yaml` of the global repository deployment in `key: otds.otdsws.serviceName`, and `<otds service port>` can be found in `key: otds.otdsws.port`.

   (Optional) Update `updateOTDScertonrestart` to true to update OTDS secret on the Documentum CM Server pod restart.

   Global repository values:

```
content-server:
    otds:
      otdsAPIsvc: otdsws:80/otdsws
      updateOTDScertonrestart: false
      auto_cert_refresh: true
      cert_jwks_url: http://otdsws:80/otdsws/oauth2/jwks
```

   Additional repository values:

```
content-server:
    otds:
      otdsAPIsvc: otdsws.namespace1:80/otdsws
      updateOTDScertonrestart: true
      auto_cert_refresh: true
      cert_jwks_url: http://otdsws.namespace1:80/otdsws/oauth2/jwks
```

9. If additional repository is deployed on the Amazon Web Services cloud platform, provide the following information if you are using static provisioning (leave these values to default in case of dynamic provisioning):

   - `existVolumePv`: Provide a unique user-defined name for PV. For example: dctmcspv.

- awsEFS: Set the value to true.
- awsEFSCSIDriver: Retain the default value as efs.csi.aws.com.
- awsEFSCSIHandle: The format is EFS file system ID1::EFS access point ID2. For example: fs-1fc8901b::fsap-0e45ed0c8888fcd34.

Externalizing Documentum CM Server

- The following configuration section will enable VM-based deployment outside the cluster to connect to Documentum CM Server. Make sure externalAccessEnabled is set to true in the common variables section in documentum/values.yaml.

```
externalAccessEnabled: &external_access_enable true
```

a. For Google Cloud Platform deployment, in the ExtCS section, set nativeExtPort as 80 and sslExtPort as 81. Keep the other default values as is.

```
ExtCS:
 tcp_route: 10.0.0.0
 nativeExtPort: 80
 sslExtPort:81
 extDbrPort: 1491
```

If the Google Cloud Platform cluster is internal or if you want to use only the internal Loadbalancer for ExtCS/ExtDocbroker, the Loadbalancer should be created internally. Set the following values in both the docbroker and content-server sections to create the internal Load Balancer service:

```
useLBAnnotations: true
LBAnnotations:
networking.gke.io/load-balancer-type: "Internal"
```

b. After the Documentum CM Server pod is deployed, it can be accessed externally.

Updates to dctm-ingress

1. Disable all services except jmsService, acsService, bpm, dsearchadminService, and indexagentService.
2. Under dctm-ingress, update ingressPrefix (prefix for the ingress name) and ingress:host.

```
dctm-ingress:
  #prefix for the ingress name
  ingressPrefix: dctm
  ingress:
    host: dctm-ingress
```

3. To enable TLS for dctm-ingress, update tls.enabled to true.

```
dctm-ingress:
  tls:
    enable: true
    secretName: *cs_secret_name
```

Note: While deploying additional repositories on Azure, make sure the ingress class is updated to the desired value and different from the global repository deployment in `documentum/platforms/azure.yaml`.

```
dctm-ingress:
  ingress:
    class: nginx2
```

Configuring object or content store for hyperscalers

1. Amazon Web Services cloud platform:

    a. If S3 store is enabled, provide the object storage details in `s3Store` in the `cs-secrets` section.

    ```
    cs-secrets:
        s3Store:
          s3StoreBaseUrl: Base_URL/bucket_name
          s3StoreCredentialID: <credentialid>
          s3StoreCredentialKEY: <credentialkey>
    ```

    b. To enable S3 store, update the `s3Store` section in the `content-server` section with `enable` as `true`. Also, set `default` as `false` to make S3 store as the non-default store. In `name`, provide the name of the store which you want to create in the S3 store. This is a user-defined field, which means the user can give any name which is not already in the S3 store. If the store configured is a public store such as Amazon S3, it is mandatory to update `proxyHost` and `proxyPort` details corresponding to the underlying S3 store. However, if the store configured is private such as NetApp, do not pass any value for these fields. Keep the default values for `proxyProtocol` and `noProxy`. `isworm` is applicable only for IBM store, therefore while configuring other stores keep the default values starting from `isworm` as follows:

    ```
    content-server:
         s3Store:
             enable: true
             default: false
             name: <s3storename>
             proxyHost: gcp-prox01-l001.otxlab.net
             proxyPort: 3128
             proxyProtocal: http
             noProxy: localhost,127.0.0.1,*.otxlab.net
             isworm: false
             vendor:
             region:
             enable_md5: true
             enable_v4signing: true
    ```

    Note: If you are using a private `s3store` created in the Amazon Web Services cloud platform, set `proxyHost` and `proxyPort` as follows:

    ```
    proxyHost: noproxy
    proxyPort: noproxy
    ```

    After the deployment is done and all the client products are configured successfully, only then the `s3store` has to be set as the default store.

2. Microsoft Azure cloud platform:

a. If Azure BLOB store is enabled, provide the object storage details in `restStore` in the `cs-secrects` section.

```
### REST Store ###
restStore:
  restStoreBaseUrl: https://dctmreststoredev.blob.core.windows.net/autotest1
  restStoreCredentialID:
  restStoreCredentialKEY:
```

Note: `restStoreBaseUrl: https://dctmreststoredev.blob.core.windows.net/autotest1` is the BLOB storage name.

b. To enable Azure BLOB store, update the `restStore` section with `enable` to `true` and provide `name`, `proxyHost` and `proxyPort` as per Azure cluster in the `content-server` section.

```
content-server:
  ### REST store enable ###
    restStore:
        enable: true
        name: autotest1
        restStoreType: 0
        proxyHost: <proxy host>
        proxyPort: <proxy post>
        proxyProtocal: http
        noProxy: localhost,127.0.0.1,*.otxlab.net
```

3. Google Cloud Platform:

   a. If Google Cloud store is enabled, get the Google Cloud Store credentials from Google Cloud Console as follows:

      i. Click on **IAM & Admin** from the left side options, select **service accounts**, click the three dots of a service account and select **Manage keys**, then create a new key by selecting **Add key option in JSON format**.

      ii. Copy the Google Cloud Store credentials from the downloaded key and populate in the `gcpStore_credentials` section.

```
### GCP Store credentials ###
gcpStore_credentials: |
{
"type": "service_account",
"project_id": "otl-eng-ecd-dctm-cs",
"private_key_id": "c8c5a09edac691d3d115b5e773ba77e99b87ac17",
"private_key": "-----BEGIN PRIVATE KEY-----
\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCvP4B4AiWahuVW
\nsM8F82V8Wf6VLiCdHSeV5TDJrY7lMaDYKb/b6qELfLcLvWtNudn7yE4NLL5g1ili
\nxRqPAeyAkRQqUhwoDEV7bHUq7qb8+fkVRYZT2NAx/
FzuVtaUEMDy69HYwSGKRdd7\n3wt51wJWIBoNn1jpEGiwTU5IPki6POOBIqkbPaMvK3233Q6MeO
4OhwS1LpQDyvZd\nOfLqXrOAc7dZOcYkQeJw
+gLIiwtbsBNDnXqoQNbJXBzhAsD9f7WuwTw8+m9MJJN3\nyyvgJjLPE3Z7w9VWmUvZLOHVL5vcN
ZClClvikXhLfMNfzmaNdZ4DTPpcOOFidxCm
\nHSN7JiJrAgMBAAECggEARI3dzpRga2ubqWbvBU/qglOeIHNz9KIkCyImDK7lOElK
\nt2TVlczaBOlbna8Sm8w5NqJrzgGTO8PTWrzqH8lL4BtgvRFzzELIp3iM+SdOkX+j
\n7atN2dz1zKgBHoc/15oaMKOJYPZkDXg2zFE7ZwUx98kFCBEusbVmjpkfJ2md/
6T7\nlcQFAbj8fjeyWbCIgOMSMvEoaO8P4Y7FiGCyno3/PoY/4iR6ewuBnBeB+6T1f/lQ
\nePvq511iXegEhZhuEcolRmBSGoWb3wABlvzyY/
K6kGpvGBtLiT3FAnTgfrpDu975\ndq8inPpVn
+ZI4CxnBHlO5AU33T5sSHN1ZZJH2T8poQKBgQDmEOOVlCzr2HTgpGML\nMy
+a7t4TeVhznbGCpOgHcArLW445jmh/x/UoWypmEydJUNRQUR8HltYFSgkadgsx
\nRtZ7PyVOrpMREz3a1+ZBooZIFKCAr6uJWcfbPdLfnU7ZqvOXfTy6pXyObGouBJrc
\nGYi6xKck6m/yU9fLO8CBaGZNewKBgQDC/
rIu9eUqvWm6oScruPYrVqROLOotSOL9\ntbP6uSkpHXE5sGgZrgO7bBrnDOAZG3TQ2qobhZEGm5
HQe9+Cbjau5HbYxJQPPYaa
\nrYyRo8u6fJGKdPymfcxOXoAsioLMvFUbyKotmDhyONLjQlMSxuwQshwdLRxOdZDh
\nkyqZ6OZTOQKBgCnSbKGfCr9gXHaNSze4+TlXnGS71RlHHAJc
+BnqAvxOSz8pJNO9\nfFH2qHFfn++SOtU6ucI+Z
+8UMy1tMcGmV8yVgOmZkEA1WDQUtNVPfstRfI+H3O2b\nVVoEFfXx
+WhVzaXcbRKcjFSzXmW5DpFdzt3sa1mph+nr1blGZ6LeYGJLAoGAGxER
\nmLUnN1SNfYtrDYWiHgfrzLKBwGHHDcKQFghnrz5X/iL/gDkJuyrZXSNfyVxnTapc
\nnG9g3yLvDZpOPv2fd41c9d/rkWX/7i6S6ZBr8hnide6hN1cU7z5C2mvrlhG6Wp3z
\nCVssOqYSl9sX/u4/zF18y4v8duOY1ccAzdZnSlECgYEAp2Og5NKJDYWfa92pUanK
\nebmR4vlDtOCV4BCs45D+d1kYpb9BoP9wDQpXnW7p7vCSrFhSOpCCn4faPd27k3WO
\nX9Bq8aV4+TQgZ8s68DN/HP93en2mYuO7NBeGfWNex51iP4BLdOaIEpnJINxYViOV\n8VfjA//
hGz3dCT2B8N/6r8s=\n-----END PRIVATE KEY-----\n",
"client_email": "138701173802-compute@developer.gserviceaccount.com",
"client_id": "117247930302497494342",
"auth_uri": "https://accounts.google.com/o/oauth2/auth",
"token_uri": "https://oauth2.googleapis.com/token",
"auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/
certs",
"client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/
x509/138701173802-compute%40developer.gserviceaccount.com"
}
```

b. If Google Cloud store is enabled, update the object storage details in `restStore` in the `cs-secrects` section.

```
### REST Store ###
restStore:
  restStoreBaseUrl: https://storage.googleapis.com/autotest1
  restStoreCredentialID:
  restStoreCredentialKEY:
## REST Store ssl certificate, below value is for (gcp blob store) change
accordnigly ###
gcpcertificate:
```

c. `restStoreBaseUrl: https://storage.googleapis.com/autotest1`. Here `autotest1` is assumed to be the cloud storage bucket name. Update `autotest1` with the bucket name. Whatever bucket name is given will be created as part of the Documentum CM Server scripts. No need to manually create the bucket in the Google cloud store.

d. Provide the valid `gcpcertificate`. It is as per Google Cloud Platform design that the public certificate expires every 90 days. After the Google Cloud Platform team releases a new certificate, the old certificate stops working even though its validity has still not expired. You must obtain a new certificate using the Java keytool utility and update the `gcpcertificate` parameter.

```
keytool -printcert -J-Dhttps.proxyHost=<proxy-host> -J-Dhttps.proxyPort=<proxy-
port> -rfc -sslServer storage.googleapis.com:443
```

Note: If the provided Google Cloud Platform certificate is valid, the storage bucket with the given name will be created as part of the Documentum CM Server scripts. No need to manually create the bucket in the Google cloud store.

e. To enable Google Cloud store, set `enable` to `true` and provide `name`, `proxyHost`, and `proxyPort` as per Google Cloud Platform cluster and set `restStoreType` to `1` in the `content-server` section.

```
content-server:
  ### REST store enable ###
    restStore:
      enable: true
      name: autotest1
      restStoreType: 1
      proxyHost: gcp-prox01-l001.otxlab.net
      proxyPort: 3128
      proxyProtocal: http
      noProxy: localhost,127.0.0.1,*.otxlab.net
```

f. To enable the Google Cloud store, set `enable` to `true`.

```
content-server:
 ### GCP store enable ###
 gcpStore:
  enable: true
```

4.7.2.5.3 Changes to dockerimages-values.yaml

1. Copy the common variables, `content-server`, `xplore`, `xda` and the `bps` sections from global repository deployment to additional repository deployment section.

2. Append the following details in the `init` container section of `content-server` for additional repository deployment. Replace `<docker image created in prerequisites>` with the Docker image created in prerequisites:

```
- name: aekkeycopy
  image: <docker image created in step 7 of prerequisites>
  imagePullPolicy: *pull_policy_type
  command: ['/bin/sh', '-c', 'yes |sudo cp -rf /aekkeypath/* /opt/dctm/shared_key']
  volumeMounts:
  - name: dcs-pg-sharedkey-pvc
    mountPath: /opt/dctm/shared_key
```

Example:

```
content-server:
  extraInitContainers:
    - name: aekkeycopy
      image: artifactory.otxlab.net/dockerhub/copyaekkey:23.4
      imagePullPolicy: *pull_policy_type
      command: ['/bin/sh', '-c', 'yes |sudo cp -rf /aekkeypath/* /opt/dctm/
shared_key']
      volumeMounts:
       - name: dcs-pg-sharedkey-pvc
         mountPath: /opt/dctm/shared_key
```

4.7.2.5.4 Changes to platforms/<platform>.yaml

1. If you are on Google Cloud Platform or Azure, update `ingress.class` with the ingress class name. Create a public IP address on Azure for `dctm-ingress` in `platforms/<platform>.yaml`.

2. Applicable for the Amazon Web Services cloud platform only: When deploying additional repository on the Amazon Web Services cloud platform, open the `aws.yaml` file in the extracted platforms folder and perform the following step: In the `alb.ingress.kubernetes.io/subnets` annotation, provide the clusters public subnet IDs separated by a comma. For example:

```
alb.ingress.kubernetes.io/subnets: subnet-0c4a1017a4ffb7962,
subnet-0bbd3ec1319a30772, subnet-0a6ab032a5e03be49
```

- If you want to use ingress in secure connection mode, do the following:

i. Generate certificate and key from a Certificate Authority and convert them to the PEM format. For example, you can use OpenSSL. The OpenSSL documentation contains detailed information.

ii. To generate the Amazon Web Services certificate Amazon Resource Name (ARN), upload the PEM files to the Amazon Web Services cloud platform. Example command:

```
C:\Users\>aws iam upload-server-certificate --server-certificate-name
<name of
certificate> --certificate-body file://<certificate in PEM format> --
privatekey
file://<key in PEM format>
When you run the preceding example command, record the ARN value in
the following output:
{
"ServerCertificateMetadata": {
"Path": "/",
"ServerCertificateName": "testcertificate",
"ServerCertificateId": "ASCA36A5J5Z4HDTEGMMXH",
"Arn": "arn:aws:iam::<AWS account ID>:server-certificate/testcertificate",
"UploadDate": "2021-11-30T10:56:25+00:00",
"Expiration": "2022-11-30T10:54:31+00:00"
}}
```

iii. Open the `aws.yaml` file in the extracted platforms folder and do the following for the ingress category `dctm-ingress`:

A. In the `alb.intress.kubernetes.io/subnets` annotation, provide the clusters' public subnet IDs separated by a comma. For example:

```
alb.ingress.kubernetes.io/subnets: subnet-0c4a1017a4ffb7962,
subnet-0bbd3ec1319a30772, subnet-0a6ab032a5e03be49
```

B. Comment the following annotation that is provided for using ingress in the non-secure connection mode:

```
alb.ingress.kubernetes.io/listen-ports: '[{"HTTP":80}]'
```

Note: For the dctm-ingress category, listen ports will be both http and https as follows. This is a requirement for xPlore:

```
alb.ingress.kubernetes.io/listen-ports: '[{"HTTP":80},{"HTTPS":443}]'
```

C. Uncomment the following annotations to use ingress in the secure connection mode:

```
alb.ingress.kubernetes.io/listen-ports: '[{"HTTPS":443}]'
alb.ingress.kubernetes.io/certificate-arn: arn:aws:iam::<AWS account
ID>:server-certificate/asbselfsigned
```

D. Update the value of `alb.ingress.kubernetes.io/group.name` to a desired value. For example:

```
alb.ingress.kubernetes.io/group.name: dctm-internal-group-2
```

Note: The value chosen as `group.name` should be unique in relation to the `group.name` used for global repository deployment and additional repository deployments.

E. Update the value of ARN in the uncommented annotation. For example:

```
alb.ingress.kubernetes.io/certificate-arn: arn:aws:iam::<AWS account
ID>:server-certificate/testcertificate
```

Note: If necessary. update the Amazon Web Services ingress annotation values in the `documentum/platforms/aws.yaml` file according to your platform's ingress.

iv. Open the `aws.yaml` file and update the IP address type used by ALB for the `appworks-gateway`, `dctm-ingress`, `vpningress`, `d2publicingress`, and `otiv` categories:

```
alb.ingress.kubernetes.io/ip-address-type: ipv4
```

The allowed values are `ipv4`, `dualstack`, and `dualstack-without-public-ipv4`.

4.7.2.6 Deploy additional repository

1. Run the following command before the deployment to validate the YAML files:

```
cd <documentum master chart directory>
helm install <documentumdeployment_name> . --values .\platforms\<platform_yaml> --
values .\dockerimages-values.yaml --values .\documentum-components.yaml --values
<resources_yaml_file>.yaml -n <namespace> --debug --dry-run
```

Note: The preceding command should not return any error.

2. Run the following Helm install command to deploy the Helm charts.

Note: Based on customer deployment size (small , medium , large, or extra), select the resources values file from the master directory and use it in the deployment command.

```
helm install <documentumdeployment_name> . --values .\platforms\<platform_yaml> --
values .\dockerimages-values.yaml --values .\documentum-components.yaml --values
<resources_yaml_file>.yaml -n <namespace>
```

4.7.2.7 Post-deployment steps

1. In global repository deployment, you must register all client webapps and their corresponding replicas (`d2config`, `d2classic`, `d2smartview` and `d2rest`) as privileged clients in all the repositories to use OTDS authentication using the following steps:

   a. Execute into client webapp using the `kubectl exec` command.

   b. Run the privileged client utility for all the repositories by updating the `<DOCBASE_NAME>` placeholder. Steps to perform:

```
kubectl exec -ti <d2 webapp>-0 -n <namespace> -- bash


java -cp $CATALINA_HOME/CustomConf:$CATALINA_HOME/webapps/<webapp folder>/WEB-
INF/lib/*:$CATALINA_HOME/webapps/<webapp folder>/utils/d2privilegedclient/*
com.opentext.d2.util.D2PrivilegedClientUtil -d <DOCBASE_NAME> -u
$INSTALL_OWNER -p $INSTALL_OWNER_PASSWORD
```

   c. Repeat the same for all webapps and repositories.

2. Follow the steps to export the login ticket from global repository deployment to additional repository deployments.

   a. Firstly exec into the Documentum CM Server pod of global repository deployment using the following command and replace `<global repository deployment namespace>` with global repository deployment:

   ```
   kubectl exec -ti dcs-pg-0 -c dcs-pg -n <global repository deployment
   namespace> -- bash
   ```

   b. Start an IAPI session by running the command. Replace `<global docbase name>` with the global repository name:

   ```
   iapi < global repository name>
   ```

   c. Run the following commands in IAPI:

   ```
               - API>
               apply,c,NULL,EXPORT_TICKET_KEY,PASSWORD,S,<password>
               ...
               q0
               API> ?,c,q0
               result
               <key_string>:
   ```

   d. Copy the ticket starting from DM_ENCR_TEXT_V2= to end of ticket string.

3. Import login ticket from global repository to additional repository.

   a. Execute into the Documentum CM Server pod of global repository deployment using the following command. Replace `<additional repository deployment namespace>` with additional repository deployment.

   ```
   kubectl exec -ti dcs-pg-0 -c dcs-pg -n <additional repository deployment
   namespace> -- bash
   ```

   b. Start an IAPI session by running the command. Replace `<additional repository name>` with additional repository name:

   ```
   iapi <additional repository name>
   ```

   c. Run the following commands in IAPI:

   ```
             - API>
               apply,c,NULL,IMPORT_TICKET_KEY,KEY_STRING,S,<ticket copied in
   previous step>,PASSWORD,S,password
               ...
               q0
               API> ?,c,q0
               result
               ------------
               T
               (1 row affected)
   ```

4. Restart the additional repository using the following commands in the same order. Replace `<additional repository name>` with the additional repository name.

```
/opt/dctm/dba/dm_shutdown_<additional repository name>

/opt/dctm/dba/dm_start_<additional repository name>
```

Note: Repeat points 2, 3, and 4 for all additional repositories if login ticket is changed in global repository deployment after the deployment on additional repository.

5. After the Documentum CM Server is deployed for the first time, Documentum CM Server can be accessed by external clients (outside the cluster) as follows:

    a. Documentum CM Server should be deployed with externalization configuration. After the initial deployment, there will be one load balancer service created for csext with an External IP.

    b. Copy the external IP and update `tcp_route` with the copied external IP and do not give any values for `nativeExtPort` or `sslExtPort`.

    ```
    ExtCS:
      tcp_route: 10.9.56.136
      nativeExtPort:
      sslExtPort:
      extDbrPort: 1491
    ```

    c. Upgrade using the following command:

    ```
    cd <Master chart directory>
    helm upgrade <documentumdeployment_name> . --values .\platforms
    \<platform_yaml> --values .\dockerimages-values.yaml --values documentum-
    components.yaml --values <resources_yaml_file>.yaml -n <namespace>
    ```

    Note: If AEK key of global repository is changed post additional repository deployment, repeat steps 5 and 7 and then step 2.

    Make sure to use a different name for the Docker image if pullpolicy for Documentum CM Server images is `IfNotPresent`.

    Perform Helm upgrade on additional repository deployment using the following command:

    ```
    cd <Master chart directory>
    helm upgrade <documentumdeployment_name> . --values .\platforms
    \<platform_yaml> --values .\dockerimages-values.yaml --values documentum-
    components.yaml --values <resources_yaml_file>.yaml -n <namespace>
    ```

    Repeat these steps for all additional repository deployments.

6. Restart the client configuration pod in global repository deployment after the deployment of additional repository. This performs the custom config import for other repositories. Otherwise, edit the global repository deployment Helm charts and perform the Helm upgrade on global repository deployment. Restart the client configuration pod in global repository deployment.

4.7.2.7.1 Additional post-deployment steps when additional repository is deployed on the Amazon Web Services cloud platform

1. Run the following command to obtain the ingress deployment addresses:

```
$ kubectl get ingress

Example output with address:

NAME                                    CLASS
HOSTS
ADDRESS
dctm-internal-ingress                   <none>
*                                                         k8s-
d2newdctmgroup-440a21863a-1248970307.us-east-1.elb.amazonaws.com
```

2. Run the following command to obtain the External IP of the csext LoadBalancer service:

```
kubectl get svc -n <namespace>

NAME            TYPE           CLUSTER-IP       EXTERNAL-
IP
PORT(S)                              AGE
csext-dcs-pg  LoadBalancer   10.100.197.39
a46526706312443188837d475c5b7e8c5-30432636.us-east-1.elb.amazonaws.com    80:30650/
TCP,81:32322/TCP             10d
dbrext-dbr    LoadBalancer   10.100.137.236
aaf2a87c480164c9cb3dd2f3167a23ff-835178087.us-east-1.elb.amazonaws.com   80:32478/
TCP,81:30552/TCP             10d
```

3. After you obtain the addresses from the ingress deployment, update the dctm-ingress host values whereever it is present in `documentum/values.yaml` with the value of corresponding ADDRESS from Step 1 of this section. Make sure you have updated the following anchor tags in `documentum/values.yaml` accordingly. For example:

```
otdsAuthSvc: &otds_auth_svc k8s-d2newdctmgroup-440a21863a-1248970307.us-
east-1.elb.amazonaws.com/otdsws
ingressDomain: &ingress_domain us-east-1.elb.amazonaws.com
```

4. Update the `tcp_route` parameter in the `ExtCS` subsection in the `content-server` section in `documentum/values.yaml` with the value of the external IP of the csext LoadBalancer service.

```
ExtCS:
  tcp_route: a46526706312443188837d475c5b7e8c5-30432636.us-east-1.elb.amazonaws.com
  nativeExtPort: 80
  sslExtPort: 81
  extDbrPort: 1491
```

5. Run the following command:

```
helm upgrade <documentumdeployment_name> . --values .\platforms\aws.yaml --values .
\dockerimages-values.yaml --values <resources_yaml_file>.yaml -n <namespace>
```

   Run the `kubectl get ingress` command again and you should see the host value updated with the corresponding address as follows:

```
$ kubectl get ingress

Example output with address:

NAME
HOSTS
```

```
ADDRESS
dctm-internal-ingress          k8s-d2newdctmgroup-440a21863a-1248970307.us-
east-1.elb.amazonaws.com    k8s-d2newdctmgroup-440a21863a-1248970307.us-
east-1.elb.amazonaws.com
```

Important

- If you are planning to use user-friendly names for the ingress host, you can map the user-friendly host names provided in the Helm charts to the ingress address in the DNS server, or you can map the user-friendly host names provided in the Helm charts to the ingress address and the host IP address in the hosts file of your client machine as applicable and access the applications using the user-friendly host names.

  To get the host IP address, run the following command:

  ```
  ping <ingress-address>
  ```

- After the Documentum CM Server pod is deployed for first time and upgraded, Documentum CM Server can be accessed by external clients (outside the cluster):

  Client connection:

  a. Documentum CM Server should be deployed and configured with external access enabled. After the initial deployment, there will be one load balancer services created for csext with an external IP.

  b. Copy `dfc.properties` from `/opt/dctm/config/dfc.properties` to your client machine from the primary Documentum CM Server pod. The original `dfc.properties` will have two host and port pairs. Remove one pair of host and port and keep one updated as follows:

  ```
   dfc.docbroker.host[0]=<External IP of dbrext LoadBalancer Service >

   dfc.docbroker.port[0]=<ExtCS.nativeExtPort>
  ```

  For example:

  ```
  dfc.docbroker.host[0]=a5e68491d32b14de4a479e41f5e3ba11-2004703172.us-
  east-1.elb.amazonaws.com
  dfc.docbroker.port[0]=80
  ```

6. In the previous deployment, if node port was used for externalizing connection broker/Documentum CM Server, if you want to use load balancer (ELB) URL, set `useELB` to `true` in `documentum/values.yaml` for the `content-server` section and upgrade the deployment. Follow step 5 of this section to upgrade the deployment with the load balancer URL.

4.7.2.7.2 OTDS configuration post-deployment

1. Access the OTDS admin website <Ingress URL>/otds-admin with admin as the user name and the value of the password you specified in OTDS Helm chart's `values.yaml` file.
2. Click **Partitions** and create a non-synchronized user partition. Add some users to this partition. After client webapp Helm charts are deployed, you can use the user account you created to log into client webapps.
3. Click **Resources** and click **Add**.
   a. Give a unique **Resource name** and **Description**.
   b. On the **Synchronization** tab, select **User and group synchronization**, **Create users and groups** and **Modify users and groups**. For **Synchronization connector**, select **REST (Generic)**.
   c. For the **Connection Information** tab, base URL should be `http://<value specified for serviceName in the Content-Server Helm chart's values.yaml>-jms-service:<value specified for ports.jmsport in Content-Server Helm chart's values.yaml>/dmotdsrest`. For example, `http://dcs-pg-jms-service:9080/dmotdsrest`. User name must be `<docbase name>\<install owner name>` and password must be `<install owner password>`. Click **Test connection**.
   d. The test should succeed.
   e. For **User Attribute Mappings**, add the client_capability attribute with `Format` value of `2`. Add the default_folder attribute with `OTDS Attribute` value as `cn` and `Format` value as `/%s`.
   f. Take the default for everything else and click **Save**.
   g. Repeat the preceding steps for all additional repositories.
4. Click **Access Roles**. You will find **Access to <your newly create resource>**.
   a. Click **Actions** and select **Include Groups**.
   b. Click **Actions** again and select **View Access Role Details**.
   c. On the **Users** tab, remove `otadmin@otds.admin` user if any. Add users for different access roles for multiple repositories.
   d. On the **Groups** tab, remove `otdsadmins@otds.admin` group if any.
   e. Click **Save**.
   f. Repeat the preceding steps for all additional repositories.
5. Click **Access Roles**. You will find Access to <global repository deployment resource>. Click **Actions** again, select **View Access Role Details** in the **Users** tab, and remove the `otadmin@otds.admin` user if any. Add the created partitions in the **Groups** tab and remove the `otdsadmins@otds.admin` group if any. Click **Save**. Repeat the steps for all additional repositories.

   Note: This step is done in order to have all the additional repository users be in global repository as well. This is necessary for client otds login in additional repositories.

6. Remove/Comment the line `X3-OTDS.defaultRepository=<docbase name>`in `d2classic-shiro-ini` ConfigMap and restart the Classic View pod in global repository deployment.

7. Allocate a license counter to the user in OTDS.

   For more information about allocating licenses, deallocating licenses, and viewing the licenses, see *OpenText Directory Services - Installation and Administration Guide (OTDS250400-IWC)*.

8. Click **Resources**.

9. Click **Actions** next to your resource and select **Consolidate** to synchronize the users to the client Documentum CM Server repository.

10. Activate the OpenText Documentum CM license. For more information, see *OpenText Documentum Content Management - Administrator User Guide (EDCAC-UGD)*.

Set the acs_base_url to localhost and acs_supported_protocol to s3 for S3 store/Google Cloud Store/ Azure BLOB store

1. Make sure `disableUpdateAcsUrl` is `true` for the `content-server` section in `documentum/values.yaml`.

2. Open IAPI in the Documentum CM Server pod and run the following command:

```
API>?,c,select object_name,r_object_id from dm_acs_config

#Note down the acs_base_url by dumping any one of the ACS r_object_id
```

3. Update `<Documentum CM Server pod name>` with the Documentum CM Server pod name and update `<cluster space>` with `env: &env_domain` mentioned in the anchor tag section of `documentum/values.yaml` and update `<original ingress URL>` with `acs_base_url` copied in the previous step.

```
API> fetch,c,<id1 returned above>

API> set,c,l,acs_base_url[0]

SET> http://<Documentum CM Server pod name>.dcs-pg.<cluster space>:9080/ACS/servlet/
ACS

API> set,c,l,acs_supported_protocol

SET> s3

API> append,c,l,acs_base_url

<original ingress URL>

API> append,c,l,acs_supported_protocol

https

API> save,c,l

...

OK
```

```
#To check if the parameter is updated, open a new iapi session and run the
following command:

API> ?,c,select object_name,r_object_id from dm_acs_config

Dump each <r_object_id> and check the acs_base_url and acs_supported_protocol
parameter

dump,c,<r_object_id>

Note: 0th index of acs_base_url and acs_supported_protocol has to be localhost and
s3 correspondingly
```

4. In case there are multiple replicas of the Documentum CM Server pod, the command `API>?,c,select object_name,r_object_id` from `dm_acs_config` returns multiple values in order of the replicas. Repeat the preceding steps for all the `r_object_ids` returned by updating the Documentum CM Server pod names. For example:

```
API> fetch,c,<id1 returned above>

API> set,c,l,acs_base_url[0]

SET> http://dcs-pg-0.dcs-pg.namespace2.svc.cluster.local:9080/ACS/servlet/ACS

API> set,c,l,acs_supported_protocol

SET> s3

API> append,c,l,acs_base_url

<original ingress URL>

API> append,c,l,acs_supported_protocol

https

API> save,c,l

...

OK
```

```
API> fetch,c,<id2 returned above>

API> set,c,l,acs_base_url[0]

SET> http://dcs-p-1.dcs-pg.namespace2.svc.cluster.local:9080/ACS/servlet/ACS

API> set,c,l,acs_supported_protocol

SET> s3

API> append,c,l,acs_base_url

<original ingress URL>

API> append,c,l,acs_supported_protocol

https

API> save,c,l

...

OK
```

Making Google Cloud store as the default store in Google Cloud Platform deployment

1. After the deployment is complete and configured with native filestore and the functional testing is completed, make the Google Cloud store as the default store using the following IAPI command in the Documentum CM Server pod:

```
kubectl exec -it dcs-pg-0 -c dcs-pg bash -n <namespace>
iapi <docbase_name>
API> ?,c,alter type dm_sysobject set DEFAULT STORAGE = '<gcpstore>'
API> ?,c,alter type dm_document set DEFAULT STORAGE = '<gcpstore>'
API> reinit,c
```

2. Steps to check the functionality of Google Cloud-object store set as the default store: Try to push/pull the content through the IAPI session using the following steps (while pushing the content, do not set `a_storage_type` as the content will by default go to the default Google Cloud store):

```
API> create,c,dm_document
...
0901e24080002dee
API> setfile,c,l,/opt/dctm/share/sample.txt,crtext
...
OK
API> save,c,l
...
OK
API> getpath,c,l
...
https://storage.googleapis.com/testidlit/01e240/80/00/11/ce.dat
API> getfile,c,l
/opt/dctm/local/process8118925741892856614.tmp/0/0601e24080009c7a.txt
```

Note: The content is now stored in the Google Cloud store https://storage.googleapis.com/testidlit/01e240/80/00/11/ce.dat and the content is in encrypted format. Any time before the DAR installation, the default store should be switched to the filestore.

Making BLOB store as the default store in Azure deployment

1. After the deployment is complete and configured with native filestore and the functional testing is completed, make the BLOB store as the default store using the following IAPI command in the Documentum CM Server pod:

```
kubectl exec -it dcs-pg-0 -c dcs-pg bash -n <namespace>
iapi <docbase_name>
API> ?,c,alter type dm_sysobject set DEFAULT STORAGE = '<Azure blob store>'
API> ?,c,alter type dm_document set DEFAULT STORAGE = '<Azure blob store>'
API> reinit,c
```

2. Steps to check the functionality of BLOB store set as the default store: Try to push/pull the content through the IAPI session using the following steps (while pushing the content, do not set `a_storage_type` as the content will by default go to the default BLOB store):

```
API> create,c,dm_document
...
0901e24080002dee
API> setfile,c,l,/opt/dctm/share/sample.txt,crtext
...
OK
```

```
API> save,c,l
...
OK
API> getpath,c,l
https://dctmreststoredev.blob.core.windows.net/testkube/01e240/80/00/15/11.dat
API> getfile,c,l  API> getfile,c,l
/opt/dctm/local/process8118925741892856614.tmp/0/0601e24080009c7a.txt
```

Note: The content is now stored in the BLOB store https://dctmreststoredev.blob.core.windows.net/testkube/01e240/80/00/15/11.dat and the content is in encrypted format. Any time before DAR installation, the default store should be switched to the filestore.

Making S3 store as the default store through IAPI in the Amazon Web Services cloud platform deployment

1. After the deployment is complete and configured with native filestore and the functional testing is completed, make the S3 store as the default store using the following IAPI command in the Documentum CM Server pod:

   ```
   kubectl exec -it cedcs-pg-0 bash -n <namespace>
   iapi <docbase_name>
   API> ?,c,alter type dm_sysobject set DEFAULT STORAGE = '<s3store>'
   API> ?,c,alter type dm_document set DEFAULT STORAGE = '<s3store>'
   API> reinit,c
   ```

2. Steps to check the functionality of S3-object store set as the default store: Try to push/pull the content through the IAPI session using the following steps (while pushing the content, do not set `a_storage_type` as the content will by default go to the default S3 store):

   ```
   API> create,c,dm_document
   ...
   0901e24080002dee
   API> setfile,c,l,/opt/dctm/share/sample.txt,crtext
   ...
   OK
   API> save,c,l
   ...
   OK
   API> getpath,c,l
   ...
   http://s3.ap-south-1.amazonaws.com/dctm-s3-bucket-1/01e240/80/00/9c/7a.dat
   API> getfile,c,l
   /opt/dctm/local/process8118925741892856614.tmp/0/0601e24080009c7a.txt
   ```

   Note: The content is now stored in S3 store http://s3.ap-south-1.amazonaws.com/dctm-s3-bucket-1/01e240/80/00/9c/7a.dat and the content is in encrypted format. Any time before DAR installation, the default store should be switched to the filestore.

After Transformation Services is configured with additional repository, restart the Documentum Administrator pod. Otherwise, the transformation feature will be disabled in Documentum Administrator until it gets restarted. Use the following steps:

```
kubectl get pod -n namespace1 | grep da
da-68f76df88b-7mxmn #da pod name
```

```
kubectl delete pod <da pod name> -n namespace1
```

4.7.2.8 Post-deployment validation

After the repositories are successfully deployed, each deployed repository must be listed in the repository list on the login page of each webapp such as `d2classic`, `d2config`, and `d2smartview`.

4.7.3 Multiple repository support in Messaging Service

For the , you must register Messaging Service and the corresponding replicas as privileged clients in all the repositories to use digital signature validation.

To register Messaging Service as privileged client:

1. Execute into Messaging Service webapp using the `kubectl exec` command.
2. Provide the appropriate value for *<DOCBASE_NAME>* and run the following command to run the privileged client utility for all the repositories:

   ```
   kubectl exec -ti <Messaging Service webapp>-0 -n <namespace> -- bash

   java -cp ${CATALINA_HOME}/webapps/dms-ws/WEB-INF/classes/:${CATALINA_HOME}/webapps/
   dms-ws/WEB-INF/lib/*:. com/documentum/dms/DmsConfigurator <DOCBASE_NAME>
   $INSTALL_OWNER_USER $INSTALL_OWNER_PASSWORD ${CATALINA_HOME}/webapps/dms-ws/WEB-INF/
   classes/
   ```

3. Repeat the preceding steps for all the replicas and repositories.

4.7.4 Multiple repository support in Documentum SAP Solutions

This section is applicable to the following components of Documentum SAP Solutions:

- Documentum Archive Services for SAP Solutions
- Documentum Archive Services for SAP Solutions ILM
- Documentum Content Services for SAP Solutions

To support multiple repositories in Documentum SAP Solutions

1. Update the Documentum Helm chart as per the instructions mentioned in "Multiple repository support in Documentum CM Server" on page 189.
2. In the `documentum-components.yaml` file, make sure the value of `global.sapconnector-init.enabled` is set to `true`.
3. In the `dockerimages-values.yaml` file, perform the following steps:

   a. If the value of `content-server.extraInitContainers.env.name` is `INSTALL_ASSAP_DARS`, then change the value of `content-server.extraInitContainers.env.value` to `true`.

b. If the value of `content-server.extraInitContainers.env.name` is `INSTALL_ILM_DARS`, then change the value of `content-server.extraInitContainers.env.value` to `true`.

c. If the value of `content-server.extraInitContainers.env.name` is `INSTALL_CSSAP_DARS`, then change the value of `content-server.extraInitContainers.env.value` to `true`.

d. If the value of `content-server.extraInitContainers.env.name` is `REINSTALL_DARS_WITH_FORCE`, then change the value of `content-server.extraInitContainers.env.value` to `true`.

4. Run the following command:

```
helm upgrade /opt/temp/documentum --values /config/configuration.yml --values /
config/constants.yaml --values /config/passwords.yaml --values /platforms/.yaml --
values /dockerimages-values.yaml --values /documentum-resources-values-test-
small.yaml --values /documentum-components.yaml --namespace
```

where <*config*> can be `extra-large`, `large`, `medium`,`medium-large`,`small`,`small-medium`, or `test-small` resource value YAML files.

The available resource YAML files contain pod sizing values, including CPU and memory.

4.8 Rotating AEK

From the 22.2 release, you can rotate AEK within the same Documentum CM Server version.

Important

- Do not change the value of any other variables in the `documentum/values.yaml` file other than the variables mentioned in this section.
- If the value of `content-server.contentserver.aek.name` in the `documentum/config/configuration.yml` file is same as the existing AEK name, then the existing AEK is reused.
- Upgrading of AEK is not supported in certificate-based communication.
- AEK rotation is supported only within the same Documentum CM Server version.

1. Modify the values of the variables depending on the following rotation requirement:

   From local to local AEK:

   - Value for `content-server.contentserver.aek.name` in the `documentum/config/configuration.yml` file.
   - (Optional) Value for `cs-secrets.contentserver.aek.algorithm` in the `documentum/config/configuration.yml` file.
   - Value for `cs-secrets.contentserver.aek.oldPassphrase` in the `documentum/config/passwords.yml` file.

- (Optional) Value for `cs-secrets.contentserver.aek.passphrase` in the `documentum/config/passwords.yml` file. Make sure that you follow the password complexity rules.

2. Run the Helm upgrade command.

4.9 Initializing Foundation Java API without dfc.properties

From the 22.4 release, you can initialize Foundation Java API without the `dfc.properties` file. To configure Foundation Java API at runtime without the use of the `dfc.properties` file, do the following:

To initialize Foundation Java API without dfc.properties:

1. Add the required Foundation Java API properties as environment variables in the Docker Compose file in the `services` section and change the values according to your requirement:

```
dfc.data.dir=/opt/dctm
dfc.docbroker.host[0]=<Connection broker IP address for Documentum CM Server>
dfc.docbroker.port[0]=<Connection broker port>
dfc.globalregistry.repository=<Global registry repository name>
dfc.globalregistry.username=dm_bof_registry
dfc.globalregistry.password=<Global registry repository password>
dfc.security.ssl.truststore=<Trust store file path>
dfc.security.ssl.truststore_password=<Trust store password>
dfc.security.ssl.use_existing_truststore=false
```

2. Call `DfPreferencesLoader.load()` before Foundation Java API initialization (that is, `getLocalClient()`).

   This initializes Foundation Java API properties entries from the environment variables instead of the `dfc.properties` file.

4.10 Decoupling product image from base Tomcat image

From the 23.2 release, decoupling of product image from the base Tomcat image is supported for the Foundation REST API and Foundation SOAP API products. From the 23.4 release, decoupling of product image from the base Tomcat image is supported for Foundation CMIS API. From the 24.2 release, decoupling of product image from the base Tomcat image is supported for Documentum Administrator and Records Manager.

The product image is decoupled from the base Tomcat image, which is built on the operating system and JDK image, resulting into the following two images:

- Base Tomcat image
- Product image

The base Tomcat image, is common for all the products, and is used to deploy the product artifacts. The product image that contains the product artifacts is implemented as an init container image.

The base Tomcat image is maintained and consumed across the products and avoids the effort to rebuild product image every time in case of any vulnerabilities reported in the base Tomcat image and the underlying operating system and JDK image. This was not possible in earlier releases until version 22.4 for Foundation REST API and Foundation SOAP API, until version 23.2 for Foundation CMIS API, and until 24.2 for Documentum Administrator and Records Manager. With the decoupling of product image from the base Tomcat image, you need to maintain only the impacted images to optimize the time cycle to replace the impacted image.

The execution and completion of the product init container image takes precedence before the execution of the base Tomcat image in the pod.

As part of the deployment of the product Helm, the init container is created and the product artifacts are pushed to a PVC. The PVC is created if there is no common PVC configuration enabled and the common PVC name is not provided in the `dctm-rest` or `dfs` or `dctm-cmis` or `da` sections in the `documentum/values.yaml` file. The product artifacts are updated in the PVC only if the existing artifacts in the PVC are older than the new artifacts that must be updated. This helps to avoid copying artifacts when pod is restarted.

As part of the deployment of the product Helm, in addition to the init container creation as previously described, the base Tomcat image container is created and all the artifacts are obtained from the PVC (including customizations, if any, is applicable only for Foundation REST API), to the base Tomcat image container. This container helps to service the product functionality.

The following tables contain more information about the Helm chart variables:

- "Global variables" on page 49 for global variables
- "Mandatory variables for Foundation CMIS API" on page 65
- "Mandatory variables for Foundation SOAP API" on page 66
- "Mandatory variables for Foundation REST API" on page 67
- "Mandatory variables for Documentum Administrator" on page 67
- "Mandatory variables for Records Client" on page 68

4.11 Logrotate

The logrotate utility is used to implement the log rotation in the Documentum CM Server pods. By default, the log rotation frequency (`logrotate.interval`) is 24 hours and this can be customized. When the size of a log file is greater than the value provided for size in `logrotate.configmap`, a backup log file is created when the log rotation frequency (`logrotate.interval`) reaches 24 hours. As and when the backup log files reaches greater than the value provided for rotate in `logrotate.configmap`, then the first backup log file gets deleted for optimization of space in the pod. This ensures that the total number of log files maintained at any time is same as the number provided for rotate in `logrotate.configmap`.

4.12 Logcleanup

The logcleanup utility is used to remove all the files from a specific folder in the Documentum CM Server pod. By default, the session logs generated in the `/opt/dctm/dba/log/<hexa_id of docbase id>/<installowner>` location are cleaned up. The cleanup interval for session logs (`logcleanup.sessionlogcleanupinterval`) is set to seven days. You can modify the cleanup interval value according to your requirement.

The log files generated in the following locations can also be cleaned up:

- `/opt/dctm/tomcat/logs`
- `/opt/dctm/product/<version>/install/logs`
- `/opt/dctm/product/<version>/thumbsrv/container/logs`
- `/opt/dctm/logs`
- `/opt/dctm_docker/logs`

To cleanup the preceding list of folders, the folder and the age of files to be removed must be mentioned in `extraEnv`.

For example, you can specify `name` and `value` for log cleanup in `extraEnv` as follows where `1` is the first folder and `2` is the age in minutes when the file is deleted:

```
- name: LOG_CLEANUP_FOLDER_1
value: "/opt/dctm/logs"
- name: LOG_CLEANUP_FOLDER_PERIOD_1
value: "2"
```

The cleanup interval (`logcleanup.interval`) for the additional folders is set to 24 hours. You can modify the cleanup interval value according to your requirement.

Note: The location provided for the logrotate and logcleanup utilities must be different.

4.13 Sensitivity labels from Microsoft Purview Information Protection

OpenText Documentum CM supports sensitivity labels from Microsoft Purview Information Protection. Sensitivity labels classify and protect Microsoft Office documents across devices, apps, and services.

OpenText Documentum CM uses sensitivity labels with and without encryption on the content to ensure the processes on the content are protected. Labels may define encryption and user access and are applied to Microsoft Office, email or PDF documents.

Documents protected by sensitivity labels from Microsoft Purview Information Protection are only supported on cloud deployments. You must enable and configure sensitivity labels from Microsoft Purview Information Protection to support processing of Microsoft Purview Information Protection protected documents.

Note: Documentum xPlore does not support sensitivity labels from Microsoft Purview Information Protection.

4.13.1 Managing documents protected by sensitivity labels in Documentum CM Server

1. Complete the app registration in the Microsoft Entra admin center.

   In addition to the app registration, do the following:

   a. On the **App registrations** page, do not specify any value for **Redirect URI (Optional)**.

   b. On the **Authentication** page, select **Accounts in this organizational directory only (Single tenant)**.

   c. On the **Configure platforms** page, select **Mobile and desktop applications**.

   d. On the **Certificates & secrets** page, configure the client secrets.

   e. On the **API permissions** page, configure the permissions as described in the following table:

API name	Permission name	Type	Admin consent required
Azure Rights Management Service	`Content.DelegatedWriter`	Application	Yes
Azure Rights Management Service	`Content.SuperUser`	Application	Yes

API name	Permission name	Type	Admin consent required
Azure Rights Management Service	`Content.Writer`	Application	Yes
Azure Rights Management Service	`user_impersonation`	Delegated	Yes
Microsoft Graph	`User.Read`	Delegated	Yes
Microsoft Information Protection (API)	`InformationProtectionPolicy.Read.All`	Application	Yes
Microsoft Information Protection Sync Service	`UnifiedPolicy.Tenant.Read`	Application	Yes

Microsoft Azure documentation contains detailed information.

2. Provide the appropriate values for the relevant variables.

   Sensitivity label variables in the documentum/config/configuration.yml file

   - `content-server.mip.proxy_host`
   - `content-server.mip.proxy_port`
   - `content-server.mip.mip_service_port`
   - `d2classic.MIPTrusted.enable`
   - `d2smartview.MIPTrusted.enable`

3. Deploy OpenText Documentum CM. For instructions, see "Deploying OpenText Documentum CM on cloud platforms" on page 45.

4. After deploying OpenText Documentum CM, provide the Microsoft Purview Information Protection configuration information in the client Admin page to create the `dmc_mip_config` object in the repository. For more information, see *OpenText Documentum Content Management - Smart View User Guide (EDCCL250400-UGD)*.

   Note: To make the Microsoft Purview Information Protection feature work, you must add the certificate from the Microsoft website to `dfc.keystore` when the Smart View is in SSL mode. Import the certificate into the `dfc.keystore` file, as specified in the `dfc.security.ssl.truststore` property in the `documentum/config/configuration.yml` file.

5. Run the following `Reinit` IAPI command on all the Documentum CM Server pods to cascade the Microsoft Purview Information Protection configuration changes to all the Documentum CM Server pods:

```
reinit,session[,server_config_name]
```

Notes

- When adding or deleting a label in the Microsoft Purview portal, you must make sure that the new or updated labels are available in the Office apps. The time taken for the new or updated labels to be available in the Office apps is approximately two hours. After the labels become available, run the `Reinit` IAPI command of all Documentum CM Server pods. You must perform this operation during maintenance mode when the load on the server is less.
- Access control with permissions, if required, must be assigned during label creation.

4.13.2 Managing documents protected by sensitivity labels in Thumbnail Server

Documents protected by sensitivity labels are processed only if:

- The processing of sensitivity labels is enabled.

  You can configure the sensitivity labels for documents on the client Admin page.
- Thumbnail Server is enabled.

Thumbnails are generated based on a specified include list. If the label is not listed in the include list, a restricted thumbnail is generated.

After a thumbnail is generated, you cannot change it from restricted to a non-restricted thumbnail or non-restricted to a restricted thumbnail by removing or adding the label from the include list. The thumbnail remains unchanged after generation.

For more information about configuring sensitivity labels, see *OpenText Documentum Content Management - Client Configuration Guide (EDCCL250400-AGD)*.

4.14 Editing ConfigMap details

ConfigMap objects are available for each webapp configuration file. More details regarding the configuration file can be found in the *OpenText Documentum Content Management client* documentation.

1. Run the following command to see a list of ConfigMap objects:

   ```
   kubectl get cm
   ```

2. Modify the configuration of a particular ConfigMap using the following command.

   ```
   kubectl get cm config_map_name -o jsonpath="{.data.config_file_name}" >
   config_file_name
   ```

3. Edit the config_file_name. Note that the file must be encoded in UTF-8.

   ```
   kubetcl create cm config_map_name --from-file config_file_name -o yaml
    --dry-run=client | kubectl apply -f -
   ```

   For example:

   ```
   kubectl get cm d2classic-logback-xml -o jsonpath="{.data.logback\.xml}"
   > logback.xml
   ```

   ```
   notepad logback.xml
   ```

   Use **File > Save As** in Notepad, and select **UTF-8** in the **Encoding** field in the Save As dialog box when saving changes to the file.

   ```
   kubectl create cm d2classic-logback-xml --from-file logback.xml -o yaml
    --dry-run=client | kubectl apply -f -
   ```

4. You can also edit the ConfigMap directly using kubectl edit cm <ConfigMap-name> command.

5. After you update the ConfigMap, wait several minutes, then check the configuration file inside the webapp container. The file content should be automatically updated by Kubernetes. Webapps will automatically pick up the changes from dfc.properties and logback.xml after a few minutes. For other configuration files, a restart of webapp might be required to pick up the changes.

The following is the list of the ConfigMaps available in a typical client deployment for the different components enabled and their purpose:

Component	ConfigMap name	Purpose
OpenText AppWorks Gateway	appworks-gateway-config-map	Configuration of the different parameters for AppWorks Gateway container.
OpenText AppWorks Gateway	tenant-config-map	Configuration of the different parameters for Appworks Gateway tenants.
OpenText AppWorks Gateway	tenant-init-container-config-map	Configuration of the different parameters for Appworks Gateway tenant init container.
OpenText AppWorks Gateway	awg-proxy-config-map	Configuration of proxy settings for Appworks Gateway tenants.
OpenText AppWorks Gateway	awg-init-container-config-map	Configuration of the different parameters for AppWorks Gateway init container.

Component	ConfigMap name	Purpose
Documentum CM Server	acs-logging-configmap	Configuration of the properties file log4j.properties for acs.log, AcsServer.log, and acs_trace.log.
Documentum CM Server	bpm-logging-configmap	Configuration of the properties file log4j.properties for bpm.log, bpm-runtime.log, and bpm_trace.log.
Documentum CM Server	cs-logging-configmap	Configuration of Foundation Java API trace, repository trace and Documentum CM Server container logs.
Documentum CM Server	d2-logback-configmap	Configuration of the properties file logback.xml for D2-JMS.log.
Classic View	d2classic-adminmessages-properties	Configuration of the adminMessages.properties file.
Classic View	d2classic-antisamy-xml	Configuration of the antisamy.xml file.
Classic View	d2classic-applicationcontext-xml	Configuration of the applicationContext.xml file.
Classic View	d2classic-bravaparameters-properties	Configuration of the brava_parameters.properties file.
Classic View	d2classic-ctfconfig-json	Configuration of the brava_parameters.properties file.
Classic View	d2classic-d2-cache-xml	Configuration of the d2-cache.xml file.
Classic View	d2classic-d2fs-properties	Configuration of the D2FS.properties file.
Classic View	d2classic-dfc-properties	Configuration of the dfc.properties file.
Classic View	d2classic-esapi-properties	Configuration of the ESAPI.properties file.
Classic View	d2classic-log4j2-properties	Configuration of the log4j2.properties file for D2-log4j.log.
Classic View	d2classic-logback-xml	Configuration of the logback.xml file for D2.log.

Component	ConfigMap name	Purpose
Classic View	d2classic-settings-properties	Configuration of the properties file settings.properties.
Classic View	d2classic-shiro-ini	Configuration of the shiro.ini file.
Classic View	d2classic-smtpc6-properties	Configuration of the smtp_c6.properties file.
Classic View	d2classic-validation-properties	Configuration of the validation.properties file.
Classic View	d2classic-server-xml	Configuration of the Tomcat server.xml file.
Classic View	d2classic-web-xml	Configuration of the Tomcat web.xml file.
client configuration	d2config-shiro-ini	Configuration of the shiro.ini file.
client configuration	d2config-d2-config-properties	Configuration of the D2-Config.properties file.
client configuration	d2config-dfc-properties	Configuration of the dfc.properties file.
client configuration	d2config-esapi-properties	Configuration of the ESAPI.properties file.
client configuration	d2config-log4j2-properties	Configuration of the log4j2.properties filr for D2-Config-log4j.log.
client configuration	d2config-logback-xml	Configuration of the logback.xml file for D2-Config.log.
client configuration	d2config-validation-properties	Configuration of the validation.properties file.
client configuration	d2config-server-xml	Configuration of the Tomcat server.xml file.
client configuration	d2config-web-xml	Configuration of the Tomcat web.xml file.
Client REST API	d2rest-antisamy-xml	Configuration of the antisamy.xml file.
Client REST API	d2rest-d2fs-properties	Configuration of the D2FS.properties file.
Client REST API	d2rest-dfc-properties	Configuration of the dfc.properties file.
Client REST API	d2rest-esapi-properties	Configuration of the ESAPI.properties file.

Component	ConfigMap name	Purpose
Client REST API	d2rest-log4j2-properties	Configuration of the log4j2.properties file for rest-api-log4j.log.
Client REST API	d2rest-logback-xml	Configuration of the logback.xml file for rest-api-logback.log.
Client REST API	d2rest-rest-api-common-ehcache-xml	Configuration of the rest-api-common-ehcache.xml file.
Client REST API	d2rest-rest-api-runtime-properties	Configuration of the rest-api-runtime.properties file.
Client REST API	d2rest-settings-properties	Configuration of the settings.properties file.
Client REST API	d2rest-trust-properties	Configuration of the trust.properties file.
Client REST API	d2rest-validation-properties	Configuration of the validation.properties file.
Client REST API	d2rest-server-xml	Configuration of the Tomcat server.xml file.
Client REST API	d2rest-web-xml	Configuration of the Tomcat web.xml file.
Smart View	d2smartview-antisamy-xml	Configuration of the antisamy.xml file.
Smart View	d2smartview-bravaparameters-properties	Configuration of the brava_parameters.properties file.
Smart View	d2smartview-d2fs-properties	Configuration of the D2FS.properties file.
Smart View	d2smartview-dfc-properties	Configuration of the dfc.properties file.
Smart View	d2smartview-esapi-properties	Configuration of the ESAPI.properties file.
Smart View	d2smartview-log4j2-properties	Configuration of the log4j2.properties file for rest-api-log4j.log.
Smart View	d2smartview-logback-xml	Configuration of the logback.xml file for D2-Smartview.log.
Smart View	d2smartview-rest-api-runtime-properties	Configuration of the rest-api-runtime.properties file.
Smart View	d2smartview-settings-properties	Configuration of the settings.properties file.

Component	ConfigMap name	Purpose
Smart View	d2smartview-sw-config-json	Configuration of the sw-config.json file.
Smart View	d2smartview-validation-properties	Configuration of the validation.properties file.
Smart View	d2smartview-server-xml	Configuration of the Tomcat server.xml file.
Smart View	d2smartview-web-xml	Configuration of the Tomcat web.xml file.
Admin Console	adminconsole-catalina-configmap	Configuration of the catalina file.
Admin Console	adminconsole-custom-certificate	Configuration of the custom certificate file.
Admin Console	adminconsole-dfc-properties	Configuration of the dfc.properties file.
Admin Console	adminconsole-log4j2-properties	Configuration of the log4j2.properties file for rest-apilog4j.log.
Admin Console	adminconsole-mailapp-properties	Configuration of the mailapp.properties file.
Admin Console	adminconsole-rest-apiruntime-properties	Configuration of the rest-apiruntime.properties file.
Admin Console	adminconsole-serviceaccount	Configuration of the service account file.
Admin Console	adminconsole-trust-properties	Configuration of the trust.properties file.
Admin Console	adminconsole-settingsproperties	Configuration of the settings.properties file.
Admin Console	adminconsole-sw-config-json	Configuration of the sw-config.json file.
Admin Console	adminconsole-server-xml	Configuration of the Tomcat server.xml file.
Admin Console	adminconsole-web-xml	Configuration of the Tomcat web.xml file.
Documentum Administrator	da-appproperties-configmap	Configuration of the properties file app.properties.
Documentum Administrator	da-logging-configmap	Configuration of the properties file log4j2.properties for documentum.log.
Documentum CM Server	dbr.configmap	Configuration of the properties file dfc.properties.

Component	ConfigMap name	Purpose
Foundation REST API	dctm-rest-configmap	Configuration of the properties file rest-api-runtime.properties.
Foundation REST API	dctm-rest-configmap-ext	Configuration of the properties file newrelic.yml.
Foundation REST API	dctm-rest-logging-configmap	Configuration of the properties file log4j2.properties for dctm-rest-services.log.
Foundation REST API	dctm-rest-rest-api-runtime-properties	Configuration of the properties file rest-api-runtime.properties.
Foundation REST API	dctm-rest-configmap-server-xml	Configuration of the Tomcat server.xml file.
Foundation REST API	dctm-rest-dfc-properties	Configuration of the dfc.properties file.
Workflow Designer	dctm-workflow-designer-dctm-workflow-designer-config	Configuration of the different parameters for Workflow Designer.
Workflow Designer	dctm-workflow-designer-dctm-workflow-designer-log-config	Configuration of the properties file log4j2.properties for WFDesigner log, WFMigration log, and WFImport log.
Foundation SOAP API	dfs-server-dfs-config	Configuration of the properties files dfc.properties & log4j.properties.
Documentum CM Server	dmotdsrest-logging-configmap	Configuration of the properties file log4j.properties for dmotdsrest.log.
Documentum CM Server	logrotate-configmap	Configuration of log rotation for catalina.out & server config log.
Documentum CM Server	new-relic.configmap	Configuration of the properties file newrelic.yml.
Documentum CM Server	oauth-logging-configmap	Configuration of the properties file log4j.properties for oauth.log.
Documentum CM Server	otdsauth-logging-configmap	Configuration of the properties file log4j.properties for otdsauth.log.

Component	ConfigMap name	Purpose
OTDS	otdsws-configmap	Configuration of the different parameters for the OTDS server.
OTDS	initdb-script	Configuration of init-db.sh for the OTDS server.
Documentum CM Server	saml-logging-configmap	Configuration of the properties file log4j.properties for SAMLAuthentication.log.
Documentum CM Server	serverapps-logging-configmap	Configuration of the properties file log4j.properties for DmMethods.log.
Documentum Records	records-appproperties-configmap	Configuration of the properties file app.properties.
Documentum Records	records-logging-configmap	Configuration of the properties file log4j2.properties for documentum.log.
Documentum Records	records-otdsproperties-configmap	Configuration of the properties file otdsoauth.properties.
OpenText Documentum CM for Microsoft 365	smartviewm365-server-xml	Configuration of the Tomcat server.xml file.
OpenText Documentum CM for Microsoft 365	smartviewm365-web-xml	Configuration of the Tomcat web.xml file.
OpenText Documentum CM for Microsoft 365	smartviewm365-launcher-json	Configuration of the properties file launcher.json.
Client REST API	d2rest-msgraph-properties	Configuration of the properties file msgraph.properties.
BPS	bps-log4j-properties-config	Configuration of the properties file log4j2.properties for bps-all.log and bps.log.
BPS	bps-bps-dfc-config	Configuration of the properties file dfc.properties.
xDA	xda-xda-config	Configuration of the properties files dfc.properties and log4j.properties for xda.log.
Foundation CMIS API	dctm-cmis-configmap	Configuration of the properties file dfc.properties for Foundation CMIS API.

Component	ConfigMap name	Purpose
Foundation CMIS API	dctm-cmis-configmap-ext	Configuration of New Relic for Foundation CMIS API.
Foundation CMIS API	dctm-cmis-logging-configmap	Configuration of the properties file log4j2.properties for dctm-cmis-services.log.
BPS	bps-bps-config	Configuration of bps config file.
Documentum Connector for Core Share	dcc-configmap	Configuration of Documentum Connector for Core Share.
Documentum Administrator	da-otdsproperties-configmap	Configuration of the properties file otdsoauth.properties.
Messaging Service	dctm-dms-configmap	Configuration of the properties file dms.properties.
Messaging Service	dctm-dms-configmap-dfc	Configuration of the properties file dfc.properties.
Messaging Service	dctm-dms-configmap-log	Configuration of the properties file log4j2.properties for DMSServer.log.
Messaging Service	dctm-dms-configmap-newrelic	Configuration of newrelic.yml.
OpenText Documentum CM for Microsoft 365	notification-service-configmap	Configuration of smartviewm365 notification service.
OpenText Intelligent Viewing	otiv-configmap	Configuration of OTIV.

Note: To add additional ConfigMap entries for webapps, use the Helm chart. For example, if you want to add entries to the ConfigMap, d2classic-bravaparameters-properties, place the entries under `d2classic.brava_parameters_properties`:

```
brava_parameters_properties: |-
               - entry1
               - entry2
               - entry3
```

Similarly, you can add extra entries to the ConfigMaps for all webapps and upgrade the deployment. Webapps pods restart automatically after an upgrade or when ConfigMaps change because of a Helm upgrade.

4.15 Integrating Core Share with Smart View

4.15.1 Prerequisites

- Access `core.opentext.com` with administrator credentials.
- On the **Security** tab, create an OAuth client by providing a description and redirect URL pointing to Smart View, for example, `http://<HOST>:<PORT>/ContextRoot` where ContextRoot can be `D2-Smarview`.
- Save the **Client ID** and **secret key**. You can use these values in `D2FS.properties` in Smart View.
- Make sure that you have enabled the Documentum Connector for Core Share component and updated all the related values with Smart View. For more information, see "Mandatory variables for Documentum Connector for Core Share" on page 83.

4.15.2 Updating Smart View in configuration.yml

You must update the `D2FS` properties in the `d2smartview` section in the `documentum/config/configuration.yml` file as follows:

```
coreOAuth2AuthUrl: https://sso.core.opentext.com/otdsws/login
coreOAuth2AccessTokenUrl: https://sso.core.opentext.com/otdsws/oauth2/token
dccUrl: <Provide the Documentum Connector for Core Share syncshare-manual/v1 end point
URL here>
```

Update the Crypto salt for client token encryption and decryption. For a multi-node deployment of REST servers, this property must be set across all REST servers. For a single-node deployment of REST servers, this property is optional. Set the value of any ASCII characters. You can specify a text no less than eight characters.

```
restApiRuntime:
          cryptoKeySalt:
```

You must update `disableUserAgents` to `OT DCC` when Core Share is used for Smart View.

```
restApiRuntime:
           disableUserAgents: OT DCC
```

If the cluster is a private cluster with no internet access to the Smart View pod then add the proxy configuration:

```
catalina_configurations:
  enable: true
  proxy:
    enable: true
```

Update the following custom properties in the d2smartview runtime properties section in values.yaml

```
restApiRuntime:
  rest_api_runtime_properties: |-
    rest.dql.access.mode=group
    rest.dql.access.groups=dmc_sync_users
```

4.15.3 Post-deployment configuration in the Smart View pod

You must update the client keystore with Documentum Connector for Core Share and Core configuration details.

1. Open the command shell in admin mode.
2. Run the following command to read the current client keystore properties and write them to a `d2keystore.properties` file. This command generates `d2keystore.properties` in the same directory.

   Note: The credentials you provide to the utility must be those of a superuser account in the global registry repository.

   If the deployment is not enabled with a vault type, run the following command:

   ```
   java -cp /opt/tomcat/CustomConf:$CATALINA_HOME/webapps/$WEBAPP_NAME/WEB-INF/lib/*:
   $CATALINA_HOME/webapps/$WEBAPP_NAME/utils/d2keystore/*
   com.emc.d2.util.D2KeyStoreUtil -docbase $DOCBASE_NAME -u $INSTALL_OWNER -p
   $INSTALL_OWNER_PASSWORD -r
   ```

   If the deployment is enabled with a vault type, then run the following command:

   ```
   java -cp /opt/tomcat/CustomConf:$CATALINA_HOME/webapps/$WEBAPP_NAME/WEB-INF/lib/*:
   $CATALINA_HOME/webapps/$WEBAPP_NAME/utils/d2keystore/*
   com.emc.d2.util.D2KeyStoreUtil -v -u $INSTALL_OWNER -docbase $DOCBASE_NAME -r
   ```

3. Edit `d2keystore.properties` and add values for the following properties:

   - dcc.user: Documentum Connector for Core Share administrator user name.
   - dcc.password: Documentum Connector for Core Share administrator password.
   - core.clientId: OAuth2 client ID registered with Core for current Smart View deployment.
   - core.clientSecret: OAuth2 client secret for current Smart View deployment.

4. In the non-vault type deployment, run the following command to read the values from `d2keystore.properties` and write them to the client keystore:

   ```
   java -cp /opt/tomcat/CustomConf:$CATALINA_HOME/webapps/$WEBAPP_NAME/WEB-INF/lib/*:
   $CATALINA_HOME/webapps/$WEBAPP_NAME/utils/d2keystore/*
   com.emc.d2.util.D2KeyStoreUtil -docbase $DOCBASE_NAME -u $INSTALL_OWNER -p
   $INSTALL_OWNER_PASSWORD -w
   ```

5. In the vault type deployment, run the following command to read the values from `d2keystore.properties` and write them to the client keystore:

   ```
   java -cp /opt/tomcat/CustomConf:$CATALINA_HOME/webapps/$WEBAPP_NAME/WEB-INF/lib/*:
   $CATALINA_HOME/webapps/$WEBAPP_NAME/utils/d2keystore/*
   com.emc.d2.util.D2KeyStoreUtil -v -u $INSTALL_OWNER -docbase $DOCBASE_NAME -w
   ```

6. Restart the Smart View pod.

4.16 Integrating Core Signature with Smart View

For information about how to integrate Core Signature with Smart View, see *OpenText Documentum Content Management - Client Configuration Guide (EDCCL250400-AGD)*.

4.17 Integrating Core Capture with Smart View

For information about how to integrate Core Capture with Smart View, see *OpenText Core Capture - Designer Guide (ECPCLCD250400-CPD)*. To know how to complete capture tasks such as scanning documents or importing files from a computer, classifying documents, and extracting data from the documents, see *OpenText Core Capture - User Guide (ECPCLCD250400-UGD)*.

4.18 Configuring Intelligent Viewing

4.18.1 Enabling Intelligent Viewing

Note: Intelligent Viewing 23.2 supports only PostgreSQL database versions 14.x and earlier. Intelligent Viewing 24.2 or later supports PostgreSQL database versions 15.x.

To enable Intelligent Viewing:

1. Update `trustedSourceOrigins` with the ingress domain URL through which the Smart View service can be accessed.

```
otiv:
  global:
    trustedSourceOrigins: https://dctm-ingress.docu.cfcr-lab.bp-paas.otlab.net
```

2. To update the client configuration objects dynamically during the Helm deployment, update the following values under the `extraEnv` section of `content-server` in the `documentum/config/configuration.yml` file:

   - Intelligent Viewing service URLs.
   - OTDS client ID and service URL.
   - Set the `UPDATE_D2CONFIG_OBJECTS` variable to `true`.
   - Set the `UPDATE_D2CONFIG_OBJECTS` variable to `false` if you do not want to update the client configuration objects dynamically or if the respective configuration objects mentioned in the following set are already updated.

```

content-server:
  extraEnv
     - name: UPDATE_D2CONFIG_OBJECTS
        value: "true"
      - name: OTIV_HIGHLIGHT_SERVICE_URL
        value: ""
     - name: OTIV_MARKUP_SERVICE_URL
        value: ""
      - name: OTIV_PUBLICATION_SERVICE_URL
          value: ""
      - name: OTIV_VIEWER_SERVICE_URL
          value: ""
      - name: OTDS_CLIENT_ID
          value: ""
      - name: OTDS_SERVICE_URL
          value: ""
```

3. If OTDS uses a private or self-signed certificate that is not trusted by root authorities, set the `enableOtivCustomizedTruststore` variable to `true` in the `otiv.global` section of `documentum/config/configuration.yml` file, and place the certificate in the `documentum/charts/otiv/certificates` folder. If the certificates directory does not exist, you must create it.

```
otiv:
  global:
     enableOtivCustomizedTruststore: true
```

4.18.2 Configuring OTDS for Intelligent Viewing

4.18.2.1 Synchronizing Intelligent Viewing users to repository

1. In the OTDS Admin Console, go to **Access Roles**.
2. Click on **Actions** of the access role and select **View access role details**.
3. Add the OAuthClients partition created by Intelligent Viewing and save the access role.
4. Go to **Resources**, select **D2 resource > Actions**, and click **Consolidate**.

4.18.2.2 Allocating the license to users

1. In the OTDS Admin Console, go to **Users and Groups**.
2. Select the required user and click **Actions > Allocate to License**.
3. From the **License** list, select **Viewing-iv**.
4. From the **Counter** list, select either **INTELLIGENT_VIEWING.FULLTIME_USERS_BASIC** or **INTELLIGENT_VIEWING.FULLTIME_USERS_REGULAR**, based per your requirement.
5. Consolidate the users.

   For more information, see “Synchronizing Intelligent Viewing users to repository” on page 234.
6. To make Intelligent Viewing work, you must perform the following license-related configuration:

   - Assign the `DOCUMENTUM.SYSTEMACCT` counter to iv-publisher@OAuthClients user.

- Assign Intelligent Viewing license to the logged in user. The logged in user must have the following licenses: OpenText Documentum CM and `INTELLIGENT_VIEWING.FULLTIME_USERS_BASIC/FULLTIME_USERS_REGULAR`.

4.18.2.3 Adding or updating oAuth clients and system attributes

If Intelligent Viewing is auto configured using the OTDS bootstrap yaml file, you can skip this entire section.

To add or update oAuth clients and system attributes:

1. In the OTDS Admin Console, go to **Oauth Clients** and select the required client (where Smart View is added as Redirect URLs).
2. Click on **Actions > Properties > Advanced**.
3. Select **Refresh token lifetime** option in the **Use session lifetime** box and provide **Access Token life time** as `1000`.
4. Add the following values in Permissible scopes, Default scopes and save the Oauth client:
   - create_publications
   - view_publications
   - view_any_publication
   - delete_any_publication
   - delete_publications
   - write_any_markups
   - read_any_markups
5. Add the following attributes in **System Config**:
   - otds.as.SameSiteCookieVal - none
   - directory.auth.EnforceSSL - false
6. Add the Smart View host to the `Trusted site` list.

4.18.3 Configuring Intelligent Viewing

After installing Intelligent Viewing and applying the OTDS licenses, configure Intelligent Viewing using client configuration. For information about configuring Intelligent Viewing, see *OpenText Documentum Content Management - Client Configuration Guide (EDCCL250400-AGD)*.

4.18.4 Importing Intelligent Viewing ingress certificate to keystore

You can add the Intelligent Viewing ingress certificate to the keystore with or without enabling SSL communication. When you enable SSL communication by setting `use_certificate` to true in `documentum/values.yaml`, import the Intelligent Viewing ingress certificate into the `dfc.keystore`.

To import the Intelligent Viewing ingress certificate:

1. Download the Intelligent Viewing ingress public certificate using one of the following options:

   - Browser

     1. Open any of Intelligent Viewing URLs and click the pad lock icon in the address bar.
     2. Click **Connection is Secure** and go to the **Details** tab.
     3. Select **Export** and save the certificate to the local system.

   - Kubernetes Secret

     1. Run the following command to get the public certificate (`tls.crt`) of the ingress from `Secret`:

        ```
        kubectl get secret <secret_name> -o yaml -n <namespace>
        ```

        You can get the secret name in `otiv.global.ingressSSLSecret` in `documentum/config/configuration.yml`

2. Run the following command to copy the certificate to Smart View pod:

   ```
   kubectl cp <localpath_to_certificate>/<certificate_name>.crt d2smartview-0:/opt/
   d2extension/d2smartview/<certificate_name>.crt -n <namespace>
   ```

3. Run the following command in the Smart View pod to import the certificate to dfc keystore:

   ```
   keytool -import -trustcacerts -file "/opt/d2extension/d2smartview/
   <certificate_name>.crt" -keystore /opt/dctm/certificate/dfc.keystore -alias
   ivpublication
   ```

   Note: As dfc.keystore is stored in PVC, restarting the pod does not reset the keystore. This is a one-time activity.

   If you have not enabled SSL, run the following command in the Smart View pod to import the certificate to java keystore:

```
keytool -import -trustcacerts -file "/opt/d2extension/d2smartview/
<certificate_name>.crt" -keystore /opt/jdk/lib/security/cacerts -alias
ivpublication
```

4. Restart the Smart View pod(s).

4.18.5 Supporting Multi-tenant OTDS for Intelligent Viewing

To support multi-tenant OTDS in Intelligent Viewing, each tenant must have its own Intelligent Viewing resource and associated license. Tenants cannot share Intelligent Viewing licenses. The resources such as users, groups, license seat reservations, and license counters are managed at the tenant level.

To support multi-tenant OTDS

1. Provide the OTDS tenant ID and tenant admin user name in the `otiv` section of `documentum/config/configuration.yml`.

   ```
   otiv:
     otdsTenantID: <tenant-id>
     otdsAdminUser: <tenant-admin-user>
   ```

2. Make sure wherever otdsws endpoint is referenced in the `documentum/values.yaml` and `documentum/config/configuration.yml` files, replace them with `https://<otds-service-internal/external-url>/otdstenant/<tenant-id>/otdsws`.

   For example

   - Replace `http://otdsws:80/otdsws` with _`http://otdsws:80/otdstenant/<tenant-id>/otdsws`.
   - Replace `https://<otds-ingress-url>/otdsws` with `https://<otds-ingress-url>/otdstenant/<tenant-id>/otdsws`.

4.19 Log management

This section provides the information about managing logs that includes collecting logs, changing log level at run time, and streaming logs to a standard output stream (`STDOUT`).

4.19.1 Collecting logs

OpenText Documentum CM provides the capability to collect both the standard output stream (`STDOUT`) and PVC-mounted log files from all the deployed pods either using the Kubernetes commands or using the log collector script file named `documentum-log-collector.sh`.

To collect logs using the log collector script file:

1. Download the `Documentum CM - Utilities.tar.gz` files from My Support and extract the contents to a temporary location.

2. From the temporary location, extract the contents of the `Documentum_Cloud_Accelerator_linux-<version>.tar.gz` file.

3. Extract the contents of the `documentum-log-collector-<version>.tar.gz` file.

4. Open a terminal window and log into the cluster using the `kubectl` command.

5. Go to the location where the Linux script file is extracted and run the following command:

   ```
   sh documentum-log-collector.sh
   ```

6. Type the Kubernetes namespace where pods are successfully deployed and press **ENTER** to list the name of the pods.

7. To download all the component and console logs for all the pods, type either `ALL` and press **ENTER** or type the number associated with the pods separated by a space and press **ENTER**.

   For example: `ALL` or `2 8 17`

   The requested pod logs are downloaded at the location where the Linux script file is available.

8. Press **ENTER** to exit.

4.19.2 Changing log level at run time

OpenText Documentum CM allows you to change the already defined log level value at run time without a manual restart of the pod.

For example, to change the defined log level for Smart View, do the following:

1. Connect to the cluster.

2. Open a command prompt window and run the following command to edit the logger ConfigMap file:

   ```
   kubectl edit configmap d2smartview-log4j2-properties
   ```

3. Change the defined value of `rootLogger.level` to a valid value.

   Valid values are `INFO`, `DEBUG`, `WARN`, `ERROR`, and `TRACE`.

4. Save and exit the logger ConfigMap file.

   The respective pod logs are updated with the new log level value within a time span of 60 seconds without restarting the pod.

4.19.3 Streaming logs to standard output

By default, all OpenText Documentum CM application logs are written in the log files. All these application logs are streamed to a standard output stream (`STDOUT`) or pod logs.

Note: For Documentum CM Server and connection broker, in addition to providing the mandatory variables as listed in "Mandatory variables for Documentum CM Server" on page 57, if you want the Documentum CM Server and connection broker logs to be streamed to a standard output or pod logs, set the value of `logsStreamToConsole` to `true` in the `documentum/values.yaml` file before you run the Helm install command as mentioned in step 5 in "To deploy OpenText Documentum CM using manual steps:" on page 45.

You can use any log collector (for example, `Fluent Bit`) to collect the logs from the pod's standard output and stream them to any log user interface (for example, `Amazon CloudWatch`).

The Fluentd sidecar is not supported for OpenText Documentum CM products and components except for Documentum CM Server.

The ConfigMap YAML files of the OpenText Documentum CM components available in the `documentum/charts` folder has the `log4j2` console appender.

Example 4-1: To stream Messaging Service logs to a standard output

The `documentum/charts/dctm-dms/templates/configMap-log.yaml` file contains the following console appender:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Values.serviceName | default .Values.global.dmsServiceName }}-configmap-log

data:
  log4j2.properties: |
    rootLogger.level = {{ .Values.log4j2.rootLogLevel}}
    rootLogger.appenderRefs = DMS_LOG, stdout
    rootLogger.appenderRef.DMS_LOG.ref = File
    rootLogger.appenderRef.stdout.ref  = STDOUT
    property.filename = /opt/tomcat/logs/
    monitorInterval = 5

    #-----------DMS-------------
    appender.DMS_LOG.type=RollingFile
    appender.DMS_LOG.name=File
    appender.DMS_LOG.fileName=/opt/tomcat/logs/DMSServer.log
    appender.DMS_LOG.filePattern=/opt/tomcat/DMSServer.log.%i.%d{yyyy-MM-dd}
    ...
    ...
    ...

    #------------------- CONSOLE -------------------------
    appender.stdout.type=Console
    appender.stdout.name=STDOUT
    appender.stdout.layout.type=PatternLayout
    appender.stdout.layout.pattern=%d{ABSOLUTE} %5p [%t] %c - %m%n
    appender.stdout.filter.threshold.type = ThresholdFilter
    appender.stdout.filter.threshold.level = DEBUG
```

```
#------------------- FILE_TRACE --------------------------
appender.FILE_TRACE.type=RollingFile
appender.FILE_TRACE.name=FileTrace
appender.FILE_TRACE.fileName=/opt/tomcat/logs/Dmslog4j2.log
appender.FILE_TRACE.filePattern=/opt/tomcat/Dmslog4j2.log.%i.%d{yyyy-MM-dd}
...
...
...
logger.rolling.appenderRef.rolling.ref = File
```

Chapter 5

Limitations and troubleshooting

This documentation provides information about limitations and troubleshooting when you deploy OpenText Documentum CM on different cloud platforms.

5.1 Limitations and troubleshooting on Microsoft Azure

This section provides information about limitations and troubleshooting on Microsoft Azure.

Table 5-1: Limitations on Microsoft Azure

Product or component name	Limitation
OpenText Documentum CM Server	• Installation path is predefined and cannot be changed. The value is `/opt/dctm/product/<product version>`. • Upgrading of schema is not supported. • When you run Tomcat on Linux, the console output is redirected to the `catalina.out` file. This is a Tomcat limitation. • Host name must have the FQDN and it must not be greater than 59 characters. • You must change the storage class according to the Azure Kubernetes service offering. The *default* storage class provisions a standard Azure disk while the *managed-premium* storage class provisions a premium Azure disk.
OpenText Documentum CM client	• Host name must have the fully qualified domain name (FQDN) and must not be greater than 59. • You must change the storage class based on the Azure Kubernetes service offering. The *default* storage class provisions a standard Azure disk while the *managed-premium* storage class provisions a premium Azure disk. • Microsoft imposes a 2 GB file size limit on content that passes through their web application firewall.

Product or component name	Limitation
OpenText Documentum CM Records Client	• Host name must have the fully qualified domain name (FQDN) and must not be greater than 59. • You must change the storage class according to the Azure Kubernetes service offering. The *default* storage class provisions a standard Azure disk while the *managed-premium* storage class provisions a premium Azure disk.

Table 5-2: Troubleshooting on Microsoft Azure

Product or component name	Problem	Cause	Solution
OpenText Documentum CM Server	When you check the status of available pods using the `kubectl get pods` command, the READY value of one or more pod(s) reads as `1/2`.	One of the two containers in the specified pod(s) is down or unavailable.	Delete the pod using the `kubectl delete pods <name of the pod>` command. The pod is recreated automatically.
OpenText Documentum CM Server	When you check the status of available deployed image, it results in the `Error: ImagePullBackOff` error.	Incorrect Helm deployment.	Delete the Helm deployment using the following command: `helm uninstall <release name> --namespace <name of namespace>` Then, provide the correct image path and redeploy the Helm chart.
OpenText Documentum CM Server, OpenText Documentum CM client, OpenText Documentum CM Records Client, OpenText Content Connect	When you configure Azure on a Linux Virtual Machine, it results in the `Unable to connect to the server` error.	Failure to connect to the server.	Use the export command as follows: `export https_proxy=<proxy_value>:<port>`

Product or component name	Problem	Cause	Solution
OpenText Documentum CM Server, OpenText Documentum CM client	When you try to describe pod, it results in the “`{chmod: changing permissions of '/var/lib/postgresql/data': Operation not permitted}`” error.	The pod does not have sufficient permissions to run privileged commands.	Provide the proper SCC to the user.
OpenText Documentum CM Server, OpenText Documentum CM client	When you run the `VolumeBinding prebind` plug-in for the pod, it results in the `Warning FailedScheduling <invalid> default-scheduler, Failed to bind volumes` error.	PVC is unable to bind.	Verify if you are able to create PVC using `assign storage class` or set the value for `pvcAccessModes` to `ReadWriteMany` and the value for PVC `size` to `256Gi`.

OpenText Documentum CM Server, OpenText Documentum CM client	When you try to deploy the pod, it results in the `Warning FailedScheduling 4m26s default-scheduler 0/5 nodes are available: 2 node(s) exceed max volume count, 3 node(s) had taint {node-role.kubernetes.io/master: }, that the pod didn't tolerate` error.	Node does not have sufficient volume disk resource. By default, OpenShift cluster creates node with size as Standard_D2s_v3 for worker node that has a maximum four disk volume capacity.	Do one of the following steps:
			• Run the following command to edit the `machinset` YAML files and deploy the size as `Standard_D32s_v 3`:
			`oc scale --replicas=0 machineset <machinesetname>`
			`oc edit machineset <machinesetname>`
			`oc scale --replicas=1 machineset <machinesetname>`
			• Increase the node resource.

Product or component name	Problem	Cause	Solution

OpenText Documentum CM Server, OpenText Documentum CM client	The `oc get` command fails.	Pending certificate signing request (CSR) for approval.	Run the following command to verify if any CSRs are pending:
			`oc get csr`
			Run the following command to approve the pending CSR (if any):
			`oc get csr -o name | xargs oc adm certificate approve`

OpenText Documentum CM Server, OpenText Documentum CM client	When you try to perform install cluster, it results in the `DEBUG Still waiting for the Kubernetes API: Get "https://api.<clustername>.<domain name>:6443/version?timeout=32s": Service Unavailable` error.	Firewall blocks the communication.	Provide access for both the port and host in the HTTPS URL.
OpenText Content Connect	When using Content Connect, Cross-Origin Resource Sharing (CORS) related errors occur even after all configurations are set.	CORS related erros occur.	• Increase the load balancer response timeout. • Add the Content Connect Admin Console URL to the `rest.cors.allowed.origins` tag in the `rest-api-runtime.properties` file.

Product or component name	Problem	Cause	Solution
OpenText Content Connect	Unable to check the New Relic logs.	Log file is not created.	Run the `docker exec -it <container name> /bin/bash` command to access the container. After the New Relic parameters are passed successfully, you can check the New Relic logs in the `newrelic_agent.log` file generated and available at the `/contentconnect` source folder. This log file helps in observing the application connectivity with the New Relic portal.
OpenText Content Connect	Cannot find the applications in the New Relic website after starting the container.	Network issues.	The New Relic SDK sends the data to the New Relic server. Fix the network issues, if any, and verify if the proxy settings are correct.

Product or component name	Problem	Cause	Solution
OpenText Documentum Connector for Core Share	In OpenText Core Share, sharing of the intended documents fail if the storage quota has exceeded.	The storage quota in the administrator account has exceeded.	The following indications are received: • The mail notification is sent to the configured Email IDs in Documentum Connector for Core Share with the following: – Subject line: `Alert: Documentum Connector for Core Share Sync Failed - Quota Limit reached for Core Share` – Message: `Hi, your quota limit is reached for Core Share account.` • On exceeding the storage limit in the administrator account of OpenText Core Share, the following log is generated in : – `syncagent.log` `*ERROR: [TasksScheduler_Worker-2]: com.emc.syncplicity.httpclient.webapi.OTCoreHttpClient - 04 Apr 2024 13:33:01,297 - status code: 413,  reason phrase: Request Entity Too Large`

Product or component name	Problem	Cause	Solution

			`org.apache.hc.client5.http.HttpResponseException: status code: 413, reason phrase: Request Entity Too Large*`
			– `syncnshareManual.log`:
			`1 INFO : [http-nio-8080-exec-67]: com.emc.syncplicity.httpclient.webapi.requests.HttpRequest - 04 Mar 2024 16:20:06,575 - HttpStatusCode : 413`
			`2 ERROR: [http-nio-8080-exec-67]: com.emc.syncplicity.connector.share.impl.OTShareOperationsUtil - 04 Mar 2024 16:20:06,609 - UploadDocumentRequest failed on CORE Wrong Server Response`
			`3 ERROR: [http-nio-8080-exec-67]: com.emc.syncplicity.connector.controller.ShareController - 04 Mar 2024 16:20:06,610 - UploadDocumentRequest failed on CORE`
			To resolve this error, you must procure additional storage quota through

Product or component name	Problem	Cause	Solution
			OpenText Core Share.

5.2 Limitations and troubleshooting on Amazon Web Services

This section provides information about limitations and troubleshooting on Amazon Web Services.

Table 5-3: Limitations on Amazon Web Services

Product or component name	Limitation
OpenText Documentum CM Server	• Installation path is predefined and cannot be changed. The value is `/opt/dctm/product/<product version>`. • Upgrading of schema is not supported. • When you run Tomcat on Linux, the console output is redirected to the `catalina.out` file. This is a Tomcat limitation. • Host name must have the FQDN and it must not be greater than 59 characters.
OpenText Documentum CM Records Client, OpenText Documentum CM client	External storage provisioners limitation: To deploy pods, you need the capability to dynamically provision both ReadWriteOnce (RWO) Persistent Volumes (PVs) and ReadWriteMany (RWX) Persistent Volumes (PVs). Cloud providers provide the built-in provisioner to dynamically provision the ReadWriteOnce Persistent Volumes. For example, in Amazon Web Services, if you specify the storageclass as `gp2` in the Persistent volume Claim (PVC), then Amazon Web Services automatically creates a Persistent Volume of requested size using Amazon Elastic Book Store (EBS). However, the default (`gp2`) does not support the ReadWriteMany operation, and hence you have to provision a Amazon EFS file system and external provisioner to dynamically manage the Persistent Volumes for ReadWriteMany Persistent Volume Claims.

Table 5-4: Troubleshooting on Amazon Web Services

Product or component name	Problem	Cause	Solution
OpenText Documentum CM Server	When you check the status of available pods using the `kubectl get pods` command, the READY value of one or more pod(s) reads as `1/2`.	One of the two containers in the specified pod(s) is down or unavailable.	Delete the pod using the `kubectl delete pods <name of the pod>` command. The pod is recreated automatically.
OpenText Documentum CM Server	When you check the status of available deployed image, it results in the `Error: ImagePullBackOff` error.	Incorrect Helm deployment.	Delete the Helm deployment using the following command: `helm uninstall <release name> --namespace <name of namespace>` Then, provide the correct image path and redeploy the Helm chart.
OpenText Content Connect	When using Content Connect, Cross-Origin Resource Sharing (CORS) related errors occur even after all configurations are set.	CORS related errors.	• Increase the Load balancer response timeout. • Add the Content Connect Admin Console URL to the `rest.cors.allowed.origins` tag in the `rest-api-runtime.properties` file.

Product or component name	Problem	Cause	Solution
OpenText Content Connect	Unable to check the New Relic logs.	Log file is not created.	Run the `docker exec -it <container name> /bin/bash` command to access the container. After the New Relic parameters are passed successfully, you can check the New Relic logs in the `newrelic_agent.log` file generated and available at the `/contentconnect` source folder. This log file helps in observing the application connectivity with the New Relic portal.
OpenText Content Connect	Cannot find the applications in the New Relic website after starting the container.	Network issues.	The New Relic SDK sends the data to the New Relic server. Fix the network issues, if any, and verify if the proxy settings are correct.

Product or component name	Problem	Cause	Solution
OpenText Documentum Connector for Core Share	In OpenText Core Share, sharing of the intended documents fail if the storage quota has exceeded.	The storage quota in the administrator account has exceeded.	The following indications are received: • The mail notification is sent to the configured Email IDs in Documentum Connector for Core Share with the following: – Subject line: `Alert: Documentum Connector for Core Share Sync Failed - Quota Limit reached for Core Share` – Message: `Hi, your quota limit is reached for Core Share account.` • On exceeding the storage limit in the administrator account of OpenText Core Share, the following log is generated in : – `syncagent.log` `*ERROR: [TasksScheduler_Worker-2]: com.emc.syncplicity.httpclient.webapi.OTCoreHttpClient - 04 Apr 2024 13:33:01,297 - status code: 413, reason phrase: Request Entity Too Large`

Product or component name	Problem	Cause	Solution

			`org.apache.hc.client5.http.HttpResponseException: status code: 413, reason phrase: Request Entity Too Large*`
			– `syncnshareManual.log`:
			`1 INFO : [http-nio-8080-exec-67]: com.emc.syncplicity.httpclient.webapi.requests.HttpRequest - 04 Mar 2024 16:20:06,575 - HttpStatusCode : 413`
			`2 ERROR: [http-nio-8080-exec-67]: com.emc.syncplicity.connector.share.impl.OTShareOperationsUtil - 04 Mar 2024 16:20:06,609 - UploadDocumentRequest failed on CORE Wrong Server Response`
			`3 ERROR: [http-nio-8080-exec-67]: com.emc.syncplicity.connector.controller.ShareController - 04 Mar 2024 16:20:06,610 - UploadDocumentRequest failed on CORE`
			To resolve this error, you must procure additional storage quota through

Product or component name	Problem	Cause	Solution
			OpenText Core Share.

5.3 Limitations and troubleshooting on Google Cloud Platform

This section provides information about limitations and troubleshooting on Google Cloud Platform.

Table 5-5: Limitations on Google Cloud Platform

Product or component name	Limitation
OpenText Documentum CM Server	• Installation path is predefined and cannot be changed. The value is `/opt/dctm/product/<product version>`. • Upgrading of schema is not supported. • When you run Tomcat on Linux, the console output is redirected to the `catalina.out` file. This is a Tomcat limitation. • Host name must have the FQDN and it must not be greater than 59 characters.
OpenText Documentum CM Records Client, OpenText Documentum CM client, OpenText Documentum CM Workflow Designer	External storage provisioners limitation. To deploy pods, you need the capability to dynamically provision both ReadWriteOnce (RWO) Persistent Volumes (PVs) and ReadWriteMany (RWX) Persistent Volumes (PVs). Cloud providers provide the built-in provisioner to dynamically provision the ReadWriteOnce PVs. For example, on Google Cloud Platform, if you specify the storageclass as standard in the Persistent Volume Claim (PVC), then Google Cloud Platform automatically creates a PV of requested size using Google Compute Engine Persistent Disk. However, the Google Compute Engine Persistent Disk does not support ReadWriteMany operation, and therefore, you have to provision a Google Cloud Filestore instance and external provisioner to dynamically manage the PVs for ReadWriteMany PVC.

Product or component name	Limitation
OpenText Documentum CM Records Client, OpenText Documentum CM Workflow Designer	To achieve ingress on Google Cloud Platform or GKE Kubernetes cluster, Google Cloud Platform Google Cloud Load Balancer (GCLB) L7 is not used as this load balancer does not communicate to the services of the ClusterIP type. Use the NGINX Ingress Controller to achieve Ingress in a GKE cluster. *Google* documentation contains detailed information.

Table 5-6: Troubleshooting on Google Cloud Platform

Product or component name	Problem	Cause	Solution
OpenText Documentum CM Server	When you check the status of available pods using the `kubectl get pods` command, the READY value of one or more pod(s) reads as `1/2`.	One of the two containers in the specified pod(s) is down or unavailable.	Delete the pod using the `kubectl delete pods <name of the pod>` command. The pod is recreated automatically.
OpenText Documentum CM Server	When you check the status of available deployed image, it results in the `Error: ImagePullBackOff` error.	Incorrect Helm deployment.	Delete the Helm deployment using the following command: `helm uninstall <release name> --namespace <name of namespace>` Then, provide the correct image path and redeploy the Helm chart.

Product or component name	Problem	Cause	Solution
OpenText Content Connect	Unable to check the New Relic logs.	Log file is not created.	Run the `docker exec -it <container name> /bin/bash` command to access the container. After the New Relic parameters are passed successfully, you can check the New Relic logs in the `newrelic_agent.log` file generated and available at the `/contentconnect` source folder. This log file helps in observing the application connectivity with the New Relic portal.
OpenText Content Connect	Cannot find the applications in the New Relic website after starting the container.	Network issues.	The New Relic SDK sends the data to the New Relic server. Fix the network issues, if any, and verify if the proxy settings are correct.

Product or component name	Problem	Cause	Solution
OpenText Documentum Connector for Core Share	In OpenText Core Share, sharing of the intended documents fail if the storage quota has exceeded.	The storage quota in the administrator account has exceeded.	The following indications are received: • The mail notification is sent to the configured Email IDs in Documentum Connector for Core Share with the following: – Subject line: `Alert: Documentum Connector for Core Share Sync Failed - Quota Limit reached for Core Share` – Message: `Hi, your quota limit is reached for Core Share account.` • On exceeding the storage limit in the administrator account of OpenText Core Share, the following log is generated in : – `syncagent.log` `*ERROR: [TasksScheduler_Worker-2]: com.emc.syncplicity.httpclient.webapi.OTCoreHttpClient - 04 Apr 2024 13:33:01,297 - status code: 413, reason phrase: Request Entity Too Large`

Product or component name	Problem	Cause	Solution

			`org.apache.hc.client5.http.HttpResponseException: status code: 413, reason phrase: Request Entity Too Large*`
			– `syncnshareManual.log`:
			`1 INFO : [http-nio-8080-exec-67]: com.emc.syncplicity.httpclient.webapi.requests.HttpRequest - 04 Mar 2024 16:20:06,575 - HttpStatusCode : 413`
			`2 ERROR: [http-nio-8080-exec-67]: com.emc.syncplicity.connector.share.impl.OTShareOperationsUtil - 04 Mar 2024 16:20:06,609 - UploadDocumentRequest failed on CORE Wrong Server Response`
			`3 ERROR: [http-nio-8080-exec-67]: com.emc.syncplicity.connector.controller.ShareController - 04 Mar 2024 16:20:06,610 - UploadDocumentRequest failed on CORE`
			To resolve this error, you must procure additional storage quota through

Product or component name	**Problem**	**Cause**	**Solution**
			OpenText Core Share.