OpenText™ Documentum™ Content Management for SAP® Solutions

Cloud Deployment Guide

This document describes how to use container, Kubernetes and Helm to deploy OpenText Documentum CM for SAP Solutions containers on a cloud platform.

EESPDC250400-ICG-EN-01

OpenText™ Documentum™ Content Management for SAP® Solutions
Cloud Deployment Guide
EESPDC250400-ICG-EN-01
Rev.: 2025-Nov-19

This documentation has been created for OpenText™ Documentum™ Content Management for SAP® Solutions 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

OpenText™ Documentum™ Content Management for SAP® Solutions integrates content and content management in leading business applications such as SAP.

OpenText provides container images that can be used to install OpenText Content Management in a Kubernetes cluster on a cloud platform, such as Microsoft Azure, Amazon Web Services, and Google Cloud Platform.

Tip: Kubernetes is a container orchestration platform for deploying and managing Docker containers. It provides many features, including automatic scaling, high availability, and fault tolerance, and simplifies the deployment.

The OpenText™ Documentum™ Content Management for SAP® Solutions deploys an OpenText Content Management image, which includes additional Content Server modules that support OpenText Content Management.

Important
Some of the products described in this guide do not support deployment on every supported cloud provider (Microsoft Azure, Amazon Web Services, and Google Cloud Platform). Refer to the release notes for details on specific product support.

The OpenText Content Management container deployments include the following:

- Two OpenText™ Content Server deployments: an Admin server (**otcs-admin**) that runs the Content Server service, agents, and the Content Server admin service, and a front-end instance (**otcs-frontend**) that runs the Content Server service, but does not run agents or the Content Server admin service.
- OpenText™ Directory Services (**otds**)
- One PostgreSQL Database servers that houses the Content Server database (**otcs-db**).

Note: OpenText recommends that you do not run your database in a container for a production deployment of OpenText Content Management. The default deployment described in this document installs database servers as containers and is suitable for a test deployment. For a production deployment, follow the instructions at “Database Servers” on page 65 to connect to an external PostgreSQL database server.

OpenText provides the container images and Helm Charts for Content Server, OpenText Directory Services. You can download them from OpenText My Support. During the deployment of Content Suite Platform described in this document, you will obtain the images for the PostgreSQL Database servers from the Docker Hub.

Important
Prior to using and configuring the software, every user on the system must be duly licensed for Content Suite Platform, OpenText Content Management and OpenText Content Management Enablers. Using this software indicates acknowledgement of this requirement and your certification that your organization is compliant with this requirement per the terms of the OpenText End User License Agreement (EULA) signed between the parties or if no such agreement is signed between the parties then per the terms of the OpenText End User License Agreement found at www.opentext.com/agreements (http://opentext.com/agreements) for the applicable region. Each user for which a product feature will be enabled is duly licensed for this functionality.

For information on specific components of OpenText Documentum CM for SAP Solutions, refer to the documentation that is available on OpenText My Support.

1.1 About this guide

This guide explains how to deploy your OpenText Documentum CM for SAP Solutions product using container images provided by OpenText. The software tools that you need to do this job are listed in "Installation requirements" on page 9. The container images and Helm charts that you need are listed in "Obtain Container Images and Helm Charts" on page 13.

You will deploy your container images in a Kubernetes cluster. Basic instructions for the creation of a Kubernetes cluster on your organization's cloud platform are provided in "Create a Kubernetes Cluster" on page 19. For more detailed information, refer to your cloud provider's documentation.

If you deploy your product using the default parameters, Content Server uses Apache Tomcat as its web server, PostgreSQL as its database. The default deployment of OpenText Documentum CM for SAP Solutions products is described in "Deploy OpenText Documentum CM for SAP Solutions" on page 29.

If the default deployment does not meet your organization's needs, you can modify the deployment of OpenText Documentum CM for SAP Solutions by using additional Helm command arguments. This is explained in "Customize the Deployment" on page 37.

Once your deployment is up and running, you can refer to "Verify the Deployment" on page 73 for basic ways to test that each product in the deployment is up and running. For more detailed information, you should refer to the Installation and Configuration guide for your product.

One of the principal benefits of containerization is ease of scaling and upgrading. These are described in "Scale the Deployment" on page 75 and "Upgrade the Deployment" on page 77.

Chapter 2

Installation requirements

To perform the steps outlined in this document, you will need the items listed below. Refer to the OpenText Content Management *Release Notes* for information on the specific platforms and versions that are supported.

Note: The instructions in this document include examples of how to use required third-party applications. To obtain additional in-depth information, consult the application's official documentation.

Docker on Linux
You will use Docker, running on a Linux computer, to tag container image files and push them to your cloud platform repository.

Kubernetes
You will use Kubernetes to run various commands during the deployment of the OpenText Documentum CM for SAP Solutions containers. Normally, you should have a local installation of Kubernetes, but it is also possible to run Kubernetes commands on certain cloud platforms. For example, on the Google Cloud Platform, you can use the Google Cloud Shell to run Kubernetes commands.

Helm
Helm is used to deploy the container images. Make sure that you use at least Helm 3.2.0. Previous versions of Helm will not work with the containerized deployment of OpenText Documentum CM for SAP Solutions.

A Cloud Platform Account
You will need a logon for a supported cloud platform where you can deploy the OpenText Documentum CM for SAP Solutions containers. OpenText supports deployment of OpenText Documentum CM for SAP Solutions containers on Microsoft Azure (https://portal.azure.com), Amazon Web Services (https://aws.amazon.com/), Google Cloud Platform (https://console.cloud.google.com) or Red Hat OpenShift Container Platform.

An SSL certificate
By default, the web applications in your deployment of OpenText Documentum CM for SAP Solutions use SSL encryption. To enable this, provide an SSL certificate (`privkey.pem` and `fullchain.pem` files) from your organization's infrastructure. You will add these files to the contents of the Helm charts that you downloaded in "The otxecm Helm Chart" on page 13, and then use them to create a Kubernetes Secret object in a Kubernetes cluster mounted on your organization's cloud platform.

For a non-production installation of OpenText Documentum CM for SAP Solutions, you can use a self-generated certificate. Free self-generated certificates are available at several sites on the Internet, including at https://letsencrypt.org.

OpenText Documentum CM for SAP Solutions Container images

OpenText provides container images for the following OpenText Documentum CM for SAP Solutions components:

- OpenText Documentum CM (running on Content Server).
- OpenText Directory Services

2.1 Microsoft Azure Installation Requirements

To deploy OpenText Documentum CM for SAP Solutions containers on Microsoft Azure, you will need everything listed in “Installation requirements“ on page 9, plus:

- An Azure account. Microsoft Azure is an ever-expanding set of cloud services to help your organization meet your business challenges.
- The Azure CLI tool. For information, see https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-apt?view=azure-cli-latest.

2.2 Amazon Web Services Installation Requirements

To deploy OpenText Documentum CM for SAP Solutions containers on Amazon Web Services (AWS), you will need everything listed in “Installation requirements“ on page 9, plus:

- An AWS account. Amazon Web Services provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis.
- The AWS CLI tool. For information, see https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html.
- The `eksctl` command-line tool. For information, see https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html.

2.3 Google Cloud Platform Installation Requirements

To deploy OpenText Documentum CM for SAP Solutions containers on Google Cloud Platform, you will need everything listed in “Installation requirements“ on page 9, plus:

- A Google Cloud Platform account. Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search.
- The Google Kubernetes Engine (GKE) is an open-source platform orchestration engine for automating deployment, scaling, and management of containerized applications. Kubernetes Engine is a standard feature of the Google Cloud

Platform. OpenText supports the use of the default version of Kubernetes that is available at the time of release on any tested platforms.

2.4 Red Hat OpenShift Installation Requirements

To deploy OpenText Documentum CM for SAP Solutions containers on Red Hat OpenShift, you will need everything listed in "Installation requirements" on page 9, plus:

- The latest version of the Red Hat OpenShift command-line interface (CLI), (the `oc` binary file).
- A Red Hat Hybrid Cloud Console logon ID.

Chapter 3

Obtain Container Images and Helm Charts

To install your OpenText Documentum CM for SAP Solutions product, you require container images and Helm charts.

You can obtain the container images from the OpenText Container Registry and the Helm charts from My Support.

3.1 OpenText Documentum CM for SAP Solutions Container Images

The container images that you use to deploy OpenText Documentum CM for SAP Solutions are available in the OpenText Container Registry and, in the case of the PostgreSQL database, on the Docker hub. The image names and tags for the pods deployed in a default deployment of OpenText Content Management can be found in the `values.yaml` file of the `otxecm` Helm chart and related charts, such as the `otds` Helm chart. You have the choice of obtaining these images manually using Docker, or configuring Helm to obtain them automatically. (See "Select your Container Registry" on page 15.)

Note: In `values.yaml`, replace `otxecm` image name with `otxecm-documentum-sap`.

3.2 The otxecm Helm Chart

The `otxecm` Helm chart orchestrates the deployment of OpenText Content Management. You can obtain it from the OpenText Helm repository. It is stored in zipped tar format, with a name similar to `otxecm-##.#.#.tgz`. Be sure to obtain the correct one for your release.

Example: The Helm Chart for OpenText Content Management 24.4 is in a file named `otxecm-24.4.0.tgz`.

Also, obtain the `otxecm-dctm-sap-yaml` sizings Helm chart from My Support.

Tip: To search the OpenText Container Registry for released `otxecm` Helm charts, first run `helm repo update` (to ensure you have an up-to-date listing of the charts that are available locally) and then use this command: `helm search repo -l opentext/otxecm`.

Obtain the otxecm Helm Chart

To obtain the `otxecm` Helm Chart, follow the steps below.

To obtain the otxecm Helm Chart

1. To add the OpenText Helm repository to Helm's list of repositories, run the following command:

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

   Replace <*user@example.com*> with your My Support logon ID, and replace <*Password*> with the password of your My Support logon ID.

2. Refresh your Helm repository information by running: `helm repo update`.

3. Obtain the version of the `otxecm` Helm chart that you require by running:

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

   Replace <*##.#.#*> with the version of the Helm chart that you require. For example:

   ```
   helm pull opentext/otxecm --version=24.4.0
   ```

   Tip: To extract the Helm chart into your current directory at the same time as you obtain it, add the `--untar` command argument. For example:

   ```
   helm pull opentext/otxecm --version=24.4.0 --untar
   ```

   If you do this, you do not need to follow the instructions in "Extract Helm Chart" on page 14.

Extract Helm Chart

Once you have obtained the Helm chart zip file, upload it to the computer that you will use to run the Helm commands. This could be your Linux computer running Docker, or it could be a cloud shell on your organization's cloud platform. Unzip the file using `gunzip` or a similar utility, and then extract its contents by running `tar xvf otxecm-<##>.<#>.<#>.tgz`.

Also, upload the obtained `otxecm-dctm-sap-yaml` sizings Helm chart to the computer and unzip the file using `gunzip` or a similar utility, and then extract the contents by running `tar xvf otxecm-dctm-sap-yaml-<##>.<#>.<#>.tgz`. Copy the standard folder in `otxecm-dctm-sap-yaml-<##>.<#>.<#>` to `otxecm` sizings folder.

When you have completed these instructions, there will be an `otxecm` subdirectory in your current directory.

3.3 Select your Container Registry

You can use the OpenText Container Registry (`registry.opentext.com`) as the source of your container images, or you can push them to your organization's container registry and use that as the source.

3.3.1 OpenText Container Registry

To use the OpenText Container Registry (`registry.opentext.com`) as your `imageSource`, create a Docker-Registry Secret and specify it as the `imagePullSecret` in your Helm deployment command. For more information, see "Docker-Registry Secret for the OpenText Container Registry" on page 29.

3.3.2 Other Container Registry

To use a different container registry, obtain the images from the OpenText Container Registry, re-tag them, and then push them to your container registry.

Notes

- In your deployment command, you will specify your Container Registry so that Helm can use the container images there to deploy OpenText Documentum CM for SAP Solutions. To enable this, you should structure your registry so that the images reside at a common location. You will specify this common location of your OpenText Documentum CM for SAP Solutions images as the value of `imageSource`.

  For example, if your Container Registry has the following folders:

  ```
  my_registry_name.my_cloud.com/my_registry/otxecm-documentum-sap
  ```

  You would specify `my_registry_name.my_cloud.com/my_registry/` as the value of `imageSource` in your Helm deployment command.

3.3.2.1 Configure Docker to access your Container Registry.

Docker needs authorization to push images to your cloud platform container registry. Refer to the section that applies to your cloud platform.

3.3.2.1.1 Configure Docker to access your Azure Container Registry

To configure Docker to access your Azure container registry, run the following command: `az acr login --name` *`<Azure_Container_Registry_Name>`*.

3.3.2.1.2 Configure Docker to access your Amazon Web Services Container Registry

Run the following commands to connect Docker and AWS CLI to your Amazon Web Services environment, including the container registry.

Connect AWS CLI to your AWS Environment

To configure AWS CLI to connect to your AWS environment, run the following command: `aws configure`. The command will prompt you to enter your ID, secret key, region, and default output format. For more information on the `aws configure` command, see the AWS help topic Configuring the AWS CLI (https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html).

Connect Docker to your AWS Container Registry

You can connect Docker to your AWS Container Registry in multiple ways. For information on AWS private registry authentication, see https://docs.aws.amazon.com/AmazonECR/latest/userguide/registry_auth.html

To use the AWS CLI to log Docker on to your AWS Container Registry, run the following command:

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

For example, if your region is `us-east-2` and your AWS Account ID is `0123456789012`, run:

```
aws ecr get-login-password --region us-east-2 | docker login --username
AWS --password-stdin 0123456789012.dkr.ecr.us-east-2.amazonaws.com
```

3.3.2.1.3 Configure Docker to access your Google Container Registry

There are several ways to configure Docker to access the Google Container Registry. The following section describes how to authorize `gcloud` as a Docker credential helper.

Tip: See https://cloud.google.com/container-registry/docs/pushing-and-pulling for more details on how to push images to Google Container Registry.

Set gcloud as a Credential Helper

To configure `gcloud` as a credential helper, run the following command on your local Linux computer:

```
gcloud auth configure-docker
```

The following output appears:

```
The following settings will be added to your Docker config file
located at [<Docker_config_file_location>]:
{
 "credHelpers": {
    "gcr.io": "gcloud",
    "us.gcr.io": "gcloud",
    "eu.gcr.io": "gcloud",
    "asia.gcr.io": "gcloud",
    "staging-k8s.gcr.io": "gcloud",
    "marketplace.gcr.io": "gcloud"

    }
}

Do you want to continue (Y/n)?
```

Click `Y` to continue. The following output confirms that the operation is successful:

```
Docker configuration file updated.
```

3.3.2.2 Obtain OpenText Container Images

Complete the following steps to obtain the OpenText container images that you require.

To obtain OpenText Container Images:

1. Log on to the OpenText Container Registry by executing the following command: `docker login registry.opentext.com`. Provide your My Support logon ID and password when you are prompted.

2. Download each image that you require by executing a `docker pull registry.opentext.com/<image_name>:<image_tag>` command.

3.3.2.3 Push Container Images

The following steps show how to prepare the OpenText Documentum CM for SAP Solutions container image and push it to your cloud platform Container Registry. Repeat these steps for OTDS image. You do not need to perform these steps for the PostgreSQL container images, because Helm will obtain them from the Docker Hub.

To push the container images to your Cloud Platform Container Registry:

1. Verify that the image has loaded successfully by running a `docker images` command. The output should list the images that you obtained in "OpenText Documentum CM for SAP Solutions Container Images" on page 13. It should appear similar to the following:

   ```
   REPOSITORY                                     TAG                 IMAGE
   ID            CREATED             SIZE
   registry.opentext.com/otxecm-documentum-sap    24.4.0
   d5a47b8d5dd3        29 hours ago        2.25GB
   ```

2. Retag the image to suit your environment by running a command similar to the following:

   `docker tag <IMAGE_ID> <My_Cloud_Repository>/otcs:24.4.0`.

   After you run the command, an additional row, similar to the following, should appear when you run a `docker images` command again.

   ```
   REPOSITORY                                       TAG                 IMAGE
   ID            CREATED            SIZE
   <My_Cloud_Repository>/otxecm-documentum-sap   24.4.0
   d5a47b8d5dd3        29 hours ago      2.25GB
   ```

3. Optional To include additional optional Content Server modules with your deployment, follow the instructions at "Add Modules, Content Server Applications, and Patches" on page 62. If you are not adding additional modules, proceed to the next step.

4. Push the image to your container registry by running a command similar to the following:

   `docker push <My_Cloud_Repository>/otxecm-documentum-sap:24.4.0`.

   You should see output that is similar to the example below.

```
The push refers to repository [<My_Cloud_Repository>/otxecm-documentum-sap]
0f5b0e3c99fb: Pushed
96f34c47d792: Pushed
32f0767c0ded: Pushed
25069c0651cb: Pushed
43e0062e571e: Pushed
492b994cb74b: Pushed
4ab452b98d33: Pushed
ded7b91e20ca: Pushed
8660aa395a36: Pushed
40d748ded39b: Pushed
6bb47c64e97b: Pushed
bb83f62cdff5: Pushed
936c79ecd43a: Pushed
13b04c8d3582: Pushed
c4d68943e210: Pushed
60cf6b6c5fdc: Pushed
bd26d4f4b2ac: Pushed
b0a0addbfee9: Pushed
37c91fb3c794: Pushed
07b6f6a2974b: Pushed
1ba63cb58be8: Pushed
latest: digest:
sha256:6087335ff1bb4f90e5dd6bd28844c96924016509449d64d949a6fa00df79a887 size: 4699
```

   Once the push completes, the image appears in your container registry.

Chapter 4

Create a Kubernetes Cluster

A Kubernetes cluster consists of at least one cluster master and multiple worker machines called nodes. These master and node machines run the Kubernetes cluster orchestration system. The Kubernetes objects that represent your containerized applications all run within a cluster.

To deploy your OpenText Documentum CM for SAP Solutions product, your Kubernetes cluster should include:

- An Ingress Controller.
- An SSL certificate to enable encryption between your OpenText Documentum CM for SAP Solutions server and the users who log on to it over the public Internet.
- DNS addresses for the public-facing components of the OpenText Documentum CM for SAP Solutions product that you deploy.

  Tip: For trial and demo deployments, a wildcard DNS address record is sufficient and can reduce the number of steps required for deploying your OpenText Documentum CM for SAP Solutions product.

The instructions in this section provide examples of how to create a Kubernetes cluster on your organization's cloud platform, and then configure it so that it is ready for the deployment of your product. For more information on creating and configuring a Kubernetes cluster on your cloud platform, refer to your cloud provider's documentation.

Tip: The instructions in this section create a Kubernetes cluster in the default namespace. To set `kubectl` to operate in a specific namespace, use the following command:

```
kubectl config set-context --current --namespace=<namespace>
```

Kubernetes namespaces are a way to divide cluster resources between multiple users. For more information on Kubernetes namespaces, see the Kubernetes official documentation (https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/).

4.1 Create a Kubernetes Cluster on Microsoft Azure

To create a Kubernetes cluster on Microsoft Azure, log on to the Azure portal (http://portal.azure.com), and then click **Kubernetes**. On the **Kubernetes services** page, click **Add** and then complete the steps in the **Create Kubernetes cluster** wizard. When you have finished, the cluster will appear on the **Kubernetes services** page.

4.1.1 Set up your Kubernetes Cluster on Microsoft Azure

The default deployment of OpenText Documentum CM for SAP Solutions on Microsoft Azure requires a specific network setup that includes a static IP address, DNS names associated with your deployment, and an SSL certificate that enables SSL communication between your deployment and the public Internet.

4.1.1.1 Log on to your Kubernetes Cluster using Azure CLI

To log Azure CLI on to your Azure Kubernetes cluster, execute the following commands:

- `az login`

  You will be prompted to open a web page in your browser and you will need to paste a code from the command output to proceed.

- `az aks get-credentials --resource-group <resource_group_name> --name <kubernetes_cluster_name>`

4.1.1.2 Attach your Container Registry to your Microsoft Azure Kubernetes Cluster

Attach your Azure container registry to your Azure Kubernetes cluster by running the following command:

```
az aks update -n <azure_cluster_name> -g <azure_resource_group> --attach-
acr <azure_container_registry>
```

4.1.1.3 Create a Static IP Address on Microsoft Azure

Create a static IP address on Microsoft Azure by running the following command:

```
az network public-ip create --name <ip_name> --resource-group <resource_
group_name> --allocation-method static --query publicIP.ipAddress --sku
Standard -o tsv
```

Tip: To display the IP address, run `az network public-ip list`.

4.1.1.4 Create DNS Services on Microsoft Azure

The Helm script that installs your OpenText Documentum CM for SAP Solutions product requires DNS names for Content Server and OpenText Directory Services. These DNS addresses will resolve to the address of the Ingress Controller that you deploy in "Deploy an Ingress Controller on Microsoft Azure" on page 21.

You can configure these DNS addresses within your organization's infrastructure or, if you prefer, you can create them on Microsoft Azure. For information on configuring DNS resolution in Azure, see https://docs.microsoft.com/en-us/azure/dns/dns-getstarted-portal.

4.1.1.5 Create an IngressSSLSecret Kubernetes Secret on Microsoft Azure

The deployment of your product requires a Kubernetes Secret object to enable encrypted communication between your deployment and the users who log on over the public Internet. To create the Secret object, copy your encryption certificate files to your Kubernetes host computer, and reference them in the following command:

```
kubectl create secret tls <secret_name> --cert <full_chain>.pem --key <private_key>.pem.
```

You will specify the name of the Kubernetes Secret as the value of `global.IngressSSLSecret` when you deploy OpenText Documentum CM for SAP Solutions. By default, the value of `global.IngressSSLSecret` is `xecm-secret`. If that is the name of your secret, you do not need to explicitly assign a value to `global.IngressSSLSecret` in your Helm deployment command.

4.1.2 Deploy an Ingress Controller on Microsoft Azure

The Helm command used to deploy your product requires an NGINX Ingress Controller to handle network traffic to and from your product's deployment.

Create an Ingress Controller that uses the static IP address that you created in "Create a Static IP Address on Microsoft Azure" on page 20. The Controller is configured to use role-based access control (`--set rbac.create=true`) and to allow users to upload files up to 1,024 MB in size (`--set controller.config.proxy-body-size=1024m`).

To create the Ingress Controller, execute a command similar to the following:

```
helm install <ingress_controller_name> ingress-nginx/ingress-nginx --set rbac.create=true --set controller.service.loadBalancerIP=<static_ip> --set controller.config.proxy-body-size=1024m
```

Tip: For information on installing NGINX Ingress Controller using Helm, see https://github.com/kubernetes/ingress-nginx/tree/master/charts/ingress-nginx.

4.2 Create a Kubernetes Cluster on Amazon Web Services

To create a Kubernetes cluster on Amazon Web Services, log on to AWS and use the `eksctl` command-line tool. After it is created, your cluster will appear on the AWS **EKS Clusters** page (https://console.aws.amazon.com/eks/home?region=<*my_aws_region*>#/clusters).

The following instructions create a Kubernetes cluster with sufficient resources to run a non-production deployment of OpenText Content Management product. For a production deployment, you should increase the resources to match the requirements of your organization. See also "Adjust Default Server Resources" on page 30.

To create a Kubernetes cluster on Amazon Web Services:

1. Log on to Amazon Web Services.
2. Obtain or create an encryption key pair that you can use with `eksctl` to create a Kubernetes cluster. For information on using AWS CLI to create a key pair, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#having-ec2-create-your-key-pair.
3. Create the Kubernetes cluster using an `eksctl` command similar to the following:

   ```
   eksctl create cluster --name <my_aws_cluster> --version 1.15 --region
   <my_aws_region> --nodegroup-name standard-workers --node-type t3.
   xlarge --nodes 3 --nodes-min 3 --nodes-max 3 --ssh-access --ssh-
   public-key <my_public_key.pem> --managed
   ```
4. To demonstrate that the cluster was created successfully, run `aws eks --region <my_aws_region> describe-cluster --name <my_aws_cluster>`.

4.2.1 Create SSL certificate, DNS names, and Ingress Controller on Amazon Web Services

The default deployment of your product on Amazon Web Services requires a specific network setup that includes an Ingress Controller, DNS names associated with your deployment, and an SSL certificate that enables encrypted communication between your deployment and the public Internet.

4.2.1.1 Add an SSL certificate on Amazon Web Services

The Helm script that deploys your product requires an SSL certificate to enable encrypted communication between your Kubernetes cluster and the users who log on over the public Internet.

Use the AWS **Certificate Manager** to add an SSL certificate to your AWS Kubernetes cluster. You can import a certificate from your organization, or you can create the certificate. Instructions for importing or creating certificates using the AWS **Certificate Manager** are available at https://docs.aws.amazon.com/acm/latest/userguide/acm-overview.html.

Once you have added the SSL certificate to AWS, expand the certificate to expose its information. Make a note of the ARN (the Amazon Resource Number) that appears in the **Details** section. You will enter the ARN as the value of `alb.ingress.kubernetes.io/certificate-arn` when you deploy the `otxecm` Helm chart.

4.2.1.2 Create an IngressSSLSecret Kubernetes Secret on Amazon Web Services

The deployment of your product requires a Kubernetes Secret object to enable encrypted communication between your deployment and the users who log on over the public Internet. To create the Secret object, copy your encryption certificate files to your Kubernetes host computer, and reference them in the following command:

```
kubectl create secret tls <secret_name> --cert <full_chain>.pem --key <private_key>.pem.
```

You will specify the name of the Kubernetes Secret as the value of `global.IngressSSLSecret` when you deploy OpenText Documentum CM for SAP Solutions. By default, the value of `global.IngressSSLSecret` is `xecm-secret`. If that is the name of your secret, you do not need to explicitly assign a value to `global.IngressSSLSecret` in your Helm deployment command.

4.2.1.3 Deploy an Ingress Controller on Amazon Web Services

Create an ALB Ingress Controller in your Kubernetes cluster to allow users to connect to your OpenText Documentum CM for SAP Solutions product over the public Internet.

Instructions for creating an ALB Ingress Controller are available at https://aws.amazon.com/blogs/opensource/kubernetes-ingress-aws-alb-ingress-controller/.

4.2.1.4 Create DNS Services on Amazon Web Services

The Helm script that installs your product requires domain names for Content Server, OpenText Directory Services, and OpenText Archive Center. These domain names will resolve to the address of the Ingress Controller that you deploy in "Deploy an Ingress Controller on Amazon Web Services" on page 23.

You can configure these domain names on AWS Route 53, within your organization's infrastructure, or elsewhere if you prefer. For information on configuring DNS resolution in AWS, consult the Amazon Web Services documentation (https://docs.aws.amazon.com/route53/?id=docs_gateway).

4.3 Create a Kubernetes Cluster on Google Cloud Platform

To create a Kubernetes cluster on Google Cloud Platform, create a Google Cloud Platform (GCP) project, add the Kubernetes engine to it, and then create the cluster. When you have finished, the cluster will appear on the **Clusters** view of the **Kubernetes Engine** page.

When you create the cluster, ensure that:

- The node pool assigned to the cluster has adequate computer resources. For a basic test deployment, three nodes with four CPUs and 15 GB of memory is sufficient. More is required for a production deployment. See also "Adjust Default Server Resources" on page 30.
- You select a GKE (Google Kubernetes Engine) **Master version** that is supported for use with OpenText Content Management product. Refer to the Release Notes for supported GKE versions.

4.3.1 Enable Kubernetes Command-Line Access

To use Kubernetes `kubectl` commands with your GCP Kubernetes cluster, you need to enable command-line access.

To connect to the Kubernetes cluster and enable Kubernetes command-line access:

1. On the **Clusters** view of the **Kubernetes Engine** page, beside the cluster that you just created, click **Connect**
2. The **Connect to the Cluster** dialog box appears, and displays the command that you need to run. To enable Kubernetes command-line access, copy this command and run it in the shell on your Kubernetes host.

   Tip: You can run this on the Google Cloud Shell if you desire. Click **Run in Cloud Shell** and the Google Cloud Shell will open with the command ready to execute.

3. To verify that Kubernetes can communicate with your cluster, run `kubectl cluster-info`. If communication is established, you should see output similar to the following:

```
$ kubectl cluster-info
Kubernetes master is running at https://<IP_address>
GLBCDefaultBackend is running at https://<IP_address>/api/v1/namespaces/kube-system/
services/default-http-backend:http/proxy
Heapster is running at https://<IP_address>/api/v1/namespaces/kube-system/services/
heapster/proxy
KubeDNS is running at https://<IP_address>/api/v1/namespaces/kube-system/services/
kube-dns:dns/proxy
Metrics-server is running at https://<IP_address>/api/v1/namespaces/kube-system/
services/https:metrics-server:/proxy
```

4.3.2 Create Static IP Address, DNS entries, and SSL certificate on Google Cloud Platform

The default deployment of OpenText Documentum CM for SAP Solutions in a Google Kubernetes Engine cluster requires a specific network setup that includes a static IP address, DNS names associated with your deployment, and an SSL certificate that enables SSL communication between your deployment and the public Internet.

4.3.2.1 Reserve a Static IP Address on Google Cloud Platform

Create a static external IP address in the same region as your Google Cloud Platform Kubernetes cluster. When you are done, make a note of the static IP address that Google Cloud Platform assigns.

To reserve a static IP address on Google Cloud Platform:

1. On the Google Cloud Platform **Navigation** menu, click **VPC Network**
2. Under **VPC network**, click **External IP addresses**.
3. On the External IP addresses page, click **RESERVE STATIC ADDRESS**.
4. On the **Reserve a static address** page, configure your static IP address:
   a. Enter a **Name** for your static IP address.
   b. Enter a **Region** for your static IP address.
   c. Click **Reserve**.

4.3.2.2 Create DNS Services on Google Cloud Platform

Create DNS names for the publicly accessible components of your deployment of OpenText Documentum CM for SAP Solutions, and map them to your static IP address.

To create DNS services on Google Cloud Platform:

1. On the Google Cloud Platform **Navigation** menu, click **Network services**
2. Under **Network services**, click **Cloud DNS**.
3. Create a Cloud DNS **Zone**.
   a. On the **Cloud DNS** page, click **CREATE ZONE**.
   b. Assign your Zone a **Zone name** and **DNS name**. For example, enter `csp.example.com`.
   c. Click **Create**.
4. Add a DNS record set for your Content Server deployment.
   a. On the **Cloud DNS** page, click the name of the DNS Zone that you created in step 3.
   b. On the **Zone details** page, click **ADD RECORD SET**.
   c. The **Create record set** page opens. The root of your DNS name appears in the **DNS Name** box.
   d. Enter the DNS name of your Content Server deployment. This is the name that users will enter in their browser to access Content Server over the public Internet. For example enter `otcs` so that the full DNS name of Content Server will be `otcs.csp.example.com`.
   e. In the **IPv4 Address** box, enter the IP address that you created in "Reserve a Static IP Address on Google Cloud Platform" on page 25.
   f. Click **Create**.
5. Repeat the above steps to create DNS record sets for OTDS.

4.3.2.3 Create an IngressSSLSecret Kubernetes Secret on Google Cloud Platform

To create the Kubernetes Secret, use the `privkey.pem` and `fullchain.pem` files from your SSL certificate.

After you complete these instructions, the Secret appears on the Google Cloud Platform Kubernetes Engine **Configuration** page. You can also display it using `kubectl get secret` and `kubectl describe secret` *`<secret_name>`*

To create a Kubernetes Secret on Google Cloud Platform:

1. Copy the certificate files to your current directory.

2. Create the Kubernetes Secret.

   `kubectl create secret tls <secret_name> --cert fullchain.pem --key privkey.pem`

4.3.3 Deploy an Ingress Controller on Google Cloud Platform

Create an Ingress Controller that uses the static IP address and DNS services that you created in "Reserve a Static IP Address on Google Cloud Platform" on page 25 and "Create DNS Services on Google Cloud Platform" on page 26. The Controller is configured to use role-based access control (`--set rbac.create=true`) and to allow users to upload files up to 1,024 MB in size (`--set controller.config.proxy-body-size=1024m`).

To create the Ingress Controller, execute a command similar to the following: `helm install <ingress_controller_name> ingress-nginx/ingress-nginx --set rbac.create=true --set controller.service.loadBalancerIP=<static_ip> --set controller.config.proxy-body-size=1024m`

Tip: For information on installing NGINX Ingress Controller using Helm, see https://github.com/kubernetes/ingress-nginx/tree/master/charts/ingress-nginx.

To verify successful deployment of the Ingress Controller, run `helm ls` or `kubectl get pods`, as in the following examples.

```
helm ls
NAME            NAMESPACE       REVISION
UPDATED                                   STATUS          CHART                   APP
VERSION
<ingress_name>  <namespace>     1               2020-03-19 10:00:01.206720631 -0400
EDT   deployed        nginx-ingress-1.31.0    0.29.0
```

```
kubectl get pods
NAME                                                            READY   STATUS
RESTARTS   AGE
<ingress_name>-ingress-nginx-controller-78bf948bc-f4gg6        1/1     Running
0          112s
```

Chapter 5

Deploy OpenText Documentum CM for SAP Solutions

To deploy your OpenText Documentum CM for SAP Solutions product on your organization's cloud platform, you will set basic deployment parameters for your cloud platform, provide an encryption certificate to enable TLS on the Ingress, adjust default server resources, and then deploy your product using a Helm command.

This section describes how to deploy your product using minimal settings: the ones that are required or recommended for initial deployment. Refer to "Customize the Deployment" on page 37 for information on additional optional settings that you can use to modify OpenText Documentum CM for SAP Solutions after it has been deployed.

5.1 Docker-Registry Secret for the OpenText Container Registry

In your Helm deployment command, specify the registry that has the container images. The OpenText Container Registry is specified by default, but you can specify a different registry.

The OpenText Container Registry requires you to log on before you can pull any container images. To enable Helm to obtain the images automatically from the OpenText Container Registry, create a Docker-Registry Secret that permits Helm to log on to the OpenText Container Registry using your My Support credentials, and specify the name of the Docker-Registry Secret in your Helm deployment command.

OpenText Container Registry

To use the OpenText Container Registry as your image source, execute the following command to create a Docker-Registry Secret, and then specify <*secret_name*> as the value of `imagePullSecret` in your Helm deployment command.

```
kubectl create secret docker-registry <secret_name> --docker-
server='https://registry.opentext.com/v1/' --docker-
username='<MySupport_logon_ID>' --docker-password='<MySupport_
password>' --docker-email='<user>@example.com'
```

Tip: Be sure to use single-quotes around your password if it has special characters in it, such as an exclamation mark (`!`).

Other Container Registry

To use a registry other than the OpenText Container Registry, follow the instructions in “Obtain Container Images and Helm Charts“ on page 13 and “Push Container Images” on page 17.

5.2 Adjust Default Server Resources

The default deployment of your product allocates sufficient compute resources for a demonstration environment. It does not provide sufficient resources for a production deployment. Before you deploy your product, review and adjust the default resource allocations for each pod.

5.2.1 OpenText Content Management front end pods

You can set resource requests and limits for CPU and memory for the pod that runs an OpenText Content Management front end instance by using the following Helm command arguments:

Note: The enforcement of resource requests and limits is enabled by default. To disable it for OpenText Content Management front ends, use the following Helm command argument:

```
--set otcs.contentServerFrontend.resources.enabled=False
```

CPU

Request

`--set otcs.contentServerFrontend.resources.requests.cpu=<#>`

The default value is `1`.

Limit

`--set otcs.contentServerFrontend.resources.limits.cpu= <#>`

The default value is `2`.

Memory

Request

`--set otcs.contentServerFrontend.resources.requests.memory=<#>`

The default value is `4Gi`.

Limit

`--set otcs.contentServerFrontend.resources.limits.memory=<#>`

The default value is `4Gi`.

5.2.2 OpenText Content Management Admin server pods

You can set resource requests and limits for CPU and memory for the pod that runs an OpenText Content Management Admin server by using the following Helm command arguments:

Note: The enforcement of resource requests and limits is enabled by default. To disable it for OpenText Content Management Admin servers, use the following Helm command argument:

```
--set otcs.contentServerAdmin.resources.enabled=False
```

CPU

Request

```
--set otcs.contentServerAdmin.resources.requests.cpu=<#>
```

The default value is `1`.

Limit

```
--set otcs.contentServerAdmin.resources.limits.cpu= <#>
```

The default value is `2`.

Memory

Request

```
--set otcs.contentServerAdmin.resources.requests.memory=<#>
```

The default value is `6Gi`.

Limit

```
--set otcs.contentServerAdmin.resources.limits.memory=<#>
```

The default value is `6Gi`.

5.2.3 OpenText Content Management Search server pods

You can set resource requests and limits for CPU and memory for the pod that runs an OpenText Content Management Search server by using the following Helm command arguments:

Note: The enforcement of resource requests and limits is enabled by default. To disable it for OpenText Content Management Admin servers, use the following Helm command argument:

```
--set otcs.contentServerBackendSearch.resources.enabled=False
```

CPU

Request

```
--set otcs.contentServerBackendSearch.resources.requests.cpu=<#>
```

The default value is `1`.

Limit

`--set otcs.contentServerBackendSearch.resources.limits.cpu= <#>`

The default value is `2`.

Memory

Request

`--set otcs.contentServerBackendSearch.resources.requests.memory=<#>`

The default value is `4Gi`.

Limit

`--set otcs.contentServerBackendSearch.resources.limits.memory=<#>`

The default value is `4Gi`.

5.3 Connect Databases

OpenText Documentum CM for SAP Solutions requires a database. Refer to "Database Servers" on page 65 for information on connecting a database to your OpenText Documentum CM for SAP Solutions.

5.4 Deploy OpenText Documentum CM for SAP Solutions Containers

This help topic explains how to deploy your product using a sample deployment command.

Tip: In most cases, to obtain the desired deployment, you will need to specify additional settings. The information in "Customize the Deployment" on page 37 explains additional Helm command arguments that you may require.

The sample command deploys OpenText Documentum CM for SAP Solutions with the following characteristics:

- Content Server is enabled.
- OTDS is enabled, providing authentication services to the other OpenText components in the deployment.
- Every pod is set to the same time zone.
- A master password is set, so that all of the users listed in the `otxecm/values.yaml` file have the same password.

Tip: Numerous additional sample deployment commands for various scenarios are available in the `readme.md` file included in the `otxecm` Helm chart.

```
helm install -i <RELEASE_NAME> otxecm \
 --set global.imageSource=<DOCKER REGISTRY PATH> \
 --set global.imagePullSecret=<SECRET NAME>
 --set global.imageSourcePublic=<DOCKER REGISTRY PATH> \
 --set global.ingressSSLSecret=<KUBERNETES SECRET NAME FOR TLS> \
 --set global.masterPassword='<PASSWORD>' \
 --set global.otcsPublicUrl="https://<OTCS URL PATH>" \
 --set global.otdsPublicUrl="https://<OTDS URL PATH>" \
 --set global.storageClassName=standard \
 --set global.storageClassNameNFS=nfs \
 --set otac.database.hostname=<DATABASE HOSTNAME> \
 --set otac.database.name=<OTAC DATABASE NAME> \
 --set otac.database.username=<OTAC DATABASE USER> \
 --set otcs.config.database.hostname=<DATABASE HOSTNAME> \
 --set otcs.config.database.name=<OTCS DATABASE NAME> \
 --set otcs.config.database.username=<OTCS DATABASE USER> \
 --set global.otac.enabled=false \
 --set global.otacc.enabled=false \
 --set global.otiv.enabled=false
 --set global.otpd.enabled=false
 --set global.otds.enabled=false
 --set otcs.image.name=otxecm-documentum-sap
 --set otcs.config.otds.serverUrl=http://<otds-service name>.<otds
namespace>.svc.cluster.local
 --set global.otxecmctrl.enabled=false
 --set global.timeZone=<TIME ZONE>
```

This sample command configures the following settings:

imageSource
: The `imageSource` property sets the location of the container registry where the `otxecm` images are stored. By default, it is `registry.opentext.com`.

  If you do not want Helm to pull the OpenText Documentum CM for SAP Solutions images from the OpenText registry, replace `registry.opentext.com` with the URL of the Docker registry that contains your OpenText Documentum CM for SAP Solutions images. (See "Select your Container Registry" on page 15.) If you leave the OpenText Container Registry as your `imageSource`, provide a value for `imagePullSecret`. See "Docker-Registry Secret for the OpenText Container Registry" on page 29.

imagePullSecret
: This is the Secret that allows Helm to log onto the OpenText Container Registry. See "Docker-Registry Secret for the OpenText Container Registry" on page 29.

imageSourcePublic
: The `imageSource` property sets the location of a container registry that stores publicly available images, for example PostgreSQL database server images. By default, it is `docker.io`.

  To use a different location for publicly available images, set `imageSourcePublic` to the address of a container registry.

IngressSSLSecret
: The `IngressSSLSecret` is the Secret that enables SSL on the Ingress. By default, the TLS secret name is `xecm-secret`. If your Secret has a different name, specify it as the value of `IngressSSLSecret`.

> **Note:** If you deploy on AWS, you must also specify the ARN of your certificate as the value of `alb.ingress.kubernetes.io/certificate-arn`. See "Add an SSL certificate on Amazon Web Services" on page 23

masterPassword
You can use the `masterPassword` property to set the password of every user in the `values.yaml` file to the same value. This is convenient in a development environment, but you should not use this in a production deployment. See "Master Password" on page 43

Public URLs (otcsPublicUrl, otdsPublicUrl, and otacPublicUrl)
The Public URLs are the web addresses of the applications in your product deployment. Set their values as follows.

- Replace `https://otcs.example.com` with the URL of your Content Server deployment.
- Replace `https://otds.example.com` with the URL of your OpenText Directory Services deployment.

  To use an OTDS server other than the one that is installed by default, enter the IP address or host name of your OTDS server instead. You will also need to modify the command arguments of the Helm command that deploys OpenText Content Management. See "OTDS" on page 69.

storageClassName and storageClassNameNFS
Default values are provided for `storageClassName` and `storageClassNameNFS`. These values must match the names of storage classes on your Kubernetes cluster, and often need to be changed. You should verify that the default values are suitable for your deployment. In many cases, they are not. For example, if you have a requirement for multi-zone failover, you probably need to specify a different value for `storageClassName` or`storageClassNameNFS`.

A value for `storageClassNameNFS` is not always required. Typically, `storageClassNameNFS` needs to be defined only if you have set the value of `otcs.config.documentStorage.type` to `efs`. The storage class specified for the `storageClassNameNFS` must have `RWX` access.

Database arguments
Content Server, OTDS, and Archive Center each require a database. For information on configuring databases for the various `otxecm` component applications, see "Database Servers" on page 65.

image.name
Name of the OpenText Documentum CM for SAP Solutions container image name.

otcs.config.otds.serverUrl
The Server URL of OTDS that is externally deployed. If OTDS is deployed in the same cluster as OpenText Documentum CM for SAP Solutions, then you may use the otds.serverUrl as `http://<otds-service name>.<otds namespace>.svc.cluster.local`.

Example: `http://otdsws.otds.svc.cluster.local`

Time zone
Use this property to set all `otxecm` components to the same time zone. See "Server Time Zone" on page 47

Once you run the Helm deployment command, you can monitor the progress of the deployment of the Kubernetes pods by executing `kubectl get pods -w` and by monitoring the deployment logs (see "Verify the Deployment" on page 73).

Chapter 6

Customize the Deployment

This chapter describes modifications that you can make to the default deployment of your OpenText Content Management product before you deploy it in the cloud using Helm.

You can modify the deployment of your product in two principal ways:

Edit the files in the Helm charts
A Helm chart is a folder hierarchy containing YAML files, XML files, and other assets. You can edit the values in these files to modify your Helm deployment.

Apply Helm command arguments
When you run your Helm deployment command, you can apply command arguments. Applying a command argument does not modify your Helm charts. It only modifies your Helm deployment. An example of a basic Helm deployment of `otxecm` is provided in "Deploy OpenText Documentum CM for SAP Solutions Containers" on page 32. There are also numerous examples of deployment commands in the `Readme.md` file included with the `otxecm` Helm chart.

OpenText recommends that, as a general rule, you do not directly edit the files in the Helm charts. Instead you should customize the deployment of your product using Helm command arguments. For that reason, this guide describes how to make modifications using Helm command arguments, wherever possible.

The majority of the changes in this section can also be made by editing the `values.yaml` file, which is included in the `otxecm` Helm chart. A structured values file, such as `values.yaml`, is a standard part of a Helm deployment. By examining the structure of the command arguments in this section, you can infer the location of the property in the `values.yaml` file and make changes to the property there, if you prefer. For more information on this topic, refer to the Helm documentation. For example, see the Helm documentation on Values Files (https://helm.sh/docs/chart_template_guide/values_files/).

6.1 Core Archive Connector

The deployment of the Core Archive Connector (`otacc`) is supported only on the OpenText Private Cloud. It is disabled by default. To explicitly disable the deployment of the Core Archive Connector, include the following command argument in your Helm deployment command:

```
--set global.otacc.enabled=false
```

6.2 IP Addressing

By default, OpenText Content Management is configured to run on an IPv4 network. You can enable it to run on IPv6 using the following command argument:

```
--set otcs.config.socketIPFamilyHint=2
```

To explicitly enable the use of IPv4, set the `FamilyHint` to 1:

```
--set otcs.config.socketIPFamilyHint=1
```

If you deploy on Amazon Web Services using an IPv6 network,, include the following additional Helm command argument

```
--set 'global.ingressAnnotations.alb\.ingress\.kubernetes\.io/ip-address-
type'='dualstack'
```

6.3 Logging

You can set the level of container deployment logging using Helm command arguments. You can also set log levels for the OpenText software applications that you deploy, using Helm command arguments or (for Content Server) an administrative settings import file.

6.3.1 Container Deployment Logging

You can configure the log levels of containers to modify the amount of information about their deployment that they log.

6.3.1.1 Content Server (otcs) Logging

To set the deployment log level for every `otcs` container in the deployment (`otcs-frontend`, `otcs-admin` and `otcs-backend-search`), use the following command argument:

`--set otcs.containerLogLevel=`*`<LogLevel>`* where <*LogLevel*> is one of DEBUG, INFO, WARNING, ERROR, or CRITICAL

If you set `otcs.containerLogLevel=DEBUG`, Content Server thread and connect logs are automatically set to level 2 (INFO). They remain at this level after deployment, until you reset them.

Tip: The `--set otcs.containerLogLevel=`*`<LogLevel>`* command sets the log level for every `otcs` container in the deployment. If you reset the log levels on the **Configure Log Settings** administration page, you do so one instance at a time. You can avoid doing this by setting log levels in an administrative settings import file, which sets levels for all of the `otcs` containers in the deployment. See "Content Server Administration Settings File" on page 48.

6.3.2 Application Logging

You can set log levels for the applications in your product deployment using the application interfaces that each product provides.

Alternatively, you can set log levels during your Helm deployment. This section explains how to set application logging levels in your container deployment command.

6.3.2.1 Content Server Logs

The Content Server log level can be set indirectly by setting the log level of the deployment of the `otcs` container, or directly by including an administrative settings import file that includes settings to configure the Content Server log level.

- If you set `otcs.containerLogLevel=DEBUG`, Content Server thread and connect logs are automatically set to level 2 (INFO). They remain at this level after deployment, until you reset them on the Content Server **Configure Log Settings** administration page. See "Content Server (otcs) Logging" on page 39.
- Using an administrative settings import file. A sample file that contains logging examples, called `adminSettings_enable_logging_example.xml`, is provided with the `otxecm` Helm chart, in the `otxecm/charts/otcs` directory. For more information on using an administrative settings import file, see "Content Server Administration Settings File" on page 48

Tip: You can combine these two methods so that Content Server thread and connect logs are automatically set to level 2 (INFO) during deployment of OpenText Documentum CM for SAP Solutions, and then reset to a different level once Content Server startup completes.

6.3.3 Fluent Bit Logging Aggregator

Fluent Bit is an open-source and multi-platform log forwarder tool which aims to be a generic Swiss knife for log collection and distribution.

If OpenText deploys OpenText Documentum CM for SAP Solutions for your organization in the OpenText Cloud, Fluent Bit is available by default for use with Content Server. You can also use it in private cloud deployments and on-premise deployments if you install and configure it yourself. Information on Fluent Bit is available at https://docs.fluentbit.io/manual/v/1.0/about.

In a containerized deployment of your product, you must configure Fluent Bit using Helm command arguments. Based on the command arguments that you provide, Helm configures Fluent Bit. It also restarts Fluent Bit and Content Server, if necessary. (If you deploy Fluent Bit on-premise, you can configure Fluent Bit on the Content Server **Configure Fluent Bit Settings** administration page.)

Content Server can send the following log types to Fluent Bit for aggregation:

Security Logs
Security Logs contain events that are of interest to security professionals. The format of the logs follows the W3C Extended Log File Format.

System Monitoring logs
System Monitoring logs provide information on the health of various Content Server components. See *OpenText Content Management Admin Help - System Administration (LLESWBA-H-AGD)*.

Tip: You can use Helm command arguments to enable these log types. See "Content Server Log Files" on page 54.

6.3.3.1 Configure OpenText Documentum CM for SAP Solutions to send logs to Fluent Bit

To configure Content Server to send supported logs to Fluent Bit, include the following command arguments in your Helm deployment command:

Enable Fluent Bit

```
--set otcs.fluentbit.enabled=true
```

Default value: `false`

Specify the location of the Fluent Bit container image

```
--set otcs.fluentbit.image.source=<Image_Location>
```

For example: `--set otcs.fluentbit.image.source=myregistry.example.net/fluentbit`

Default value: `docker.io`

Specify the name of the Fluent Bit container image

```
--set otcs.fluentbit.image.name=<Fluent_Bit_Image_Name>
```

For example: `--set otcs.fluentbit.image.name=fluent/fluent-bit`

Default value: `fluent/fluent-bit`

Specify the tag of the Fluent Bit container image

`--set otcs.fluentbit.image.tag=`*`<Fluent_Bit_Image_Tag>`*

For example: `--set otcs.fluentbit.image.tag=1.8`

Default value:`1.8`

Specify the logs that Fluent Bit will receive from Content Server

`--set otcs.fluentbit.logsToMonitor[`*`<#>`*`]="`*`<Log_Type>`*`"`

Content Server supports sending Security Logs and System Monitoring logs to Fluent Bit.

Security Logs

`--set otcs.fluentbit.logsToMonitor[`*`<#>`*`]="security"`

System Monitoring logs

`--set otcs.fluentbit.logsToMonitor[`*`<#>`*`]="sysmon"`

6.3.3.2 Configure Fluent Bit proxy settings

If the Fluent Bit server is accessed through a proxy server, include any of the following Helm command arguments that you require.

Enable the use of a proxy server

`--set otcs.fluentbit.proxy.enabled=true`

Proxy server connection details

To send supported logs to a remote Fluent Bit server, use the following Helm command arguments.

Specify the IP address or host name of the proxy server.

`--set otcs.fluentbit.proxy.host=`*`<Proxy_Host_Name>`*

Specify the port of the proxy server.

`--set otcs.fluentbit.proxy.port=`*`<Proxy_Port>`*

If the proxy server requires authentication, include the following command arguments:

`--set otcs.fluentbit.proxy.enableauthentication=true`

`--set otcs.fluentbit.proxy.username=`*`<User_Name>`*

`--set otcs.fluentbit.proxy.password=`*`<Password>`*

6.3.3.3 Configure Fluent Bit custom output

You can configure Fluent Bit to send output to another application. For example, you can configure Fluent Bit to send data to Splunk for analysis.

To configure Fluent Bit to send output to a another application:

- Include a configuration file in the `fluentbit` subfolder of the `otcs` Helm chart. The file must contain the configuration data that the other application requires to allow Fluent Bit to connect to it.
- Enable Fluent Bit custom output, and refer to the custom output configuration file using the following Helm command arguments:
  - `--set otcs.fluentbit.customOutput.enabled=true`
  - `--set otcs.fluentbit.customOutput.customOutputFilePath=`*`<configuration_file>`*

For example, to configure Fluent Bit to send output to a Splunk instance located at the IP address, `10.0.0.5`:

- Create a file called `customoutput.conf` in the `fluentbit` subfolder of the `otcs` Helm chart, with the following contents:

```
[OUTPUT]
        Name Splunk
        Host 10.0.0.5
        Port 8080
        Match *
        Splunk_Token ########-####-####-####-###########
        Splunk_Send_Raw on
```

- Include the following command arguments in your Helm deployment command:
  - `--set otcs.fluentbit.customOutput.enabled=true`
  - `--set otcs.fluentbit.customOutput.customOutputFilePath=customoutput.conf3`

6.4 Passwords and sensitive data

There are several ways that you can set passwords for your deployment at the time you run your Helm command:

- Set a Master Password for the deployment. See “Master Password” on page 43.
- Use suitable Helm command arguments. See “Helm Command Arguments for Individual Passwords” on page 43
- Create Kubernetes Secrets that contain passwords. See “Kubernetes Secrets for Passwords and Sensitive Data” on page 45.

Notes

- Any password that you specify must conform to the default password complexity rules in OTDS: it must have at least eight characters, and it must include one uppercase letter, one lowercase letter, one digit, and one symbol.
- Use of a Master Password can simplify the deployment of a non-production environment, but is not recommended for production environments.
- OpenText recommends the use of Kubernetes Secret for passwords management in a production environment.
- You can set different passwords using Helm command arguments and Kubernetes Secrets concurrently, but if you set the same password using both methods, the password set in the Kubernetes Secret is the one that is applied.

6.4.1 Master Password

The master password, if set, is applied to all of the users listed in the `otxecm/values.yaml` file.

Use of a master password can simplify the deployment of a non-production environment, but is not recommended for production environments. To set a master password, use the `--set global.masterPassword='<PASSWORD>'` Helm command argument.

For example, to set `MyP@ssw0rd` as the master password, include the following command argument in your Helm deployment command:

```
--set global.masterPassword='MyP@ssw0rd'
```

6.4.2 Helm Command Arguments for Individual Passwords

The users that are listed in the `values.yaml` file fulfill varying roles in a deployment of OpenText Documentum CM for SAP Solutions. You can set their passwords individually by applying a Helm command argument when you deploy your product.

Note: In some cases, the same user password may need to be updated in more than one command argument.

This section explains how to change the passwords of the users listed in the `values.yaml` file of the `otxecm` Helm chart. (You can also change the *names* of users in the `values.yaml` file, but this is not normally advisable or necessary.) This section of the documentation is organized by `otxecm` component. It lists users and passwords in the order that they are found in the `values.yaml` file.

This

6.4.2.1 Passwords in the Directory Services (otds) chart.

The password of the OTDS administrator is set in the `otds` chart

OpenText Directory Services Administrator

This is the OTDS administrator (`admin`), who configures OTDS and Content Server.

To change the password of the OTDS administrator, use the Helm command argument `--set otds.password='<password>'`

Note: The OTDS administrator must have the same password as the Content Server admin user.

6.4.2.2 Passwords in the Content Server (otcs) chart.

The passwords of the Content Server Admin user, the PostgreSQL system user, and the Content Server database user are set in the `otcs` chart.

6.4.2.2.1 Content Server Admin user

This is the Content Server administrative user that exists in every Content Server deployment.

To change the password of the Content Server Admin user, use the Helm command argument `--set otcs.passwords.adminPassword='<password>'`.

Notes

- The Content Server admin user must have the same password as the OTDS administrator. (See "OpenText Directory Services Administrator" on page 44.)

6.4.2.2.2 System User for the Content Server Database Server

This is the administrative user of the Content Server Database Server that performs the setup of the Content Server database.

To change the password of the system user for the Content Server database server, use the Helm command argument `--set otcs.passwords.database.adminPassword='<password>'`

6.4.2.2.3 Content Server Database User

This is the Content Server database user (`cs`) that interacts with the Content Server database

To change the password of the Content Server database user, use the Helm command argument `--set otcs.passwords.database.Password='<password>'`.

6.4.3 Kubernetes Secrets for Passwords and Sensitive Data

A Secret is a Kubernetes object that contains sensitive data such as passwords, tokens, and keys. Such information might otherwise be put in a Pod specification or in a container image. Using a Secret means that you do not need to include confidential code in your deployment files. Bear in mind, however, that anyone with access to your Kubernetes cluster may be able to access the content of your Secrets.

OpenText recommends the use of Kubernetes Secrets for individual passwords in a production deployment of OpenText Documentum CM for SAP Solutions.

To set a password in a Kubernetes Secret for a user in the `otxecm` deployment, edit the `otxecm/example-secret.yaml` file that is included with the `otxecm` Helm chart. The passwords in this file are organized in sections that correspond to the charts used by the `otxecm` Helm chart. For example, the passwords that are listed in the section that begins with `## otcs` correspond to values that are by default set in the `otcs` (Content Server) chart.

If you set one password in the `otxecm/example-secret.yaml` file, Kubernetes will obtain *all* of the passwords listed in it from Secrets that you generate from the file. For that reason, if you set one password in the `otxecm/example-secret.yaml` file, you must uncomment and set all of the other passwords that your deployment uses. You do not need to set passwords for `otxecm` components that you do not use, however. So, for example, if you deploy an `otcs` Content Server container, you must provide values for every password in the `otcs` section.

Passwords that are set in Kubernetes Secrets must be maintained in Kubernetes Secrets. Once you have set a password using a Kubernetes Secret, you cannot update it using Helm command arguments or any application user interface, including the Content Server user interface.

6.4.3.1 Configure Secrets for the passwords of your system users

The `otxecm/example-secret.yaml` file contains passwords of technical users involved in the `otxecm` deployment, and access keys for connecting to hyperscalers. For any user password that you would like to change, uncomment the line and replace `your-base64-encoded-password` with the base64-encoded value of your password.

To set system user passwords in Kubernetes Secrets, follow the instructions below. When you deploy your OpenText Content Management product, Helm will set the passwords that are contained in your Kubernetes Secrets.

To set system-user passwords using Secrets:

1. Base64-encode the password that you wish to use. You can use the Linux command `echo -n 'myPassword' | base64 -w 0`.

2. Edit the `otxecm/example-secret.yaml` file. Uncomment the name of any passwords that you want to change and replace `your-base64-encoded-password` with the string that you generated in step 1 above.

   For example, to change the password of the `ADMIN_USER_PASSWORD` with the base64-encoded value of `myPassword`, modify the line as follows:

   ```
   ## ADMIN_USER_PASSWORD determines the admin user and admin server password. This
   must match OTDS_PASS below.
   ADMIN_USER_PASSWORD: bXlQYXNzd29yZA==
   ```

   Tip: As the comments in the `example-secret.yaml` indicate, `ADMIN_USER_PASSWORD` and `OTDS_PASS` must have the same value.

3. Generate a Kubernetes Secret based on the contents of the `example-secret.yaml` file, as follows:

   `kubectl create -f otxecm/example-secret.yaml`

   This command creates a Kubernetes Secret named `otxecm-secrets`.

4. Use the following Helm command argument to reference the Kubernetes Secret in your Helm deployment command: `--set global.existingSecret=<Secret>`

When you deploy your OpenText Content Management product, the passwords will be set during the deployment.

6.5 Pod Annotations

You can apply annotations to the pods deployed by the `otcs` Helm sub-chart.

To apply pod annotations to `otcs` pods, use the following Helm command argument:

```
--set otcs.podAnnotations.<key>="<value>"
```

For example to set the annotation, `test: otxecmtest`, include the command argument: `--set otcs.podAnnotations.test="otxecmtest"`.

6.6 Server Time Zone

By default, the operating system is set to the UTC time zone (`Etc/UTC`) in all of the pods in your deployment of OpenText Documentum CM for SAP Solutions.

To set a different time zone that applies to every pod in the deployment, use the `--set global.timeZone=<Time_Zone>` command argument. If necessary, you can also set the time zone of each individual pod separately.

6.6.1 Modify the Time Zone of every Pod

To have the pods run in a time zone of your choice, set the value of `timeZone` to a valid time zone from the IANA time zone database.

For example, to set the pods to run in the time zone of Dubai (Gulf Standard Time), include the following command: `--set global.timeZone=Asia/Dubai`

6.6.2 Modify the Time Zone of Individual Pods

The value of `global.timeZone` is applied to every pod in the deployment. They all run in the same time zone. If you need to modify the time zone of individual pods, use the following command arguments when you deploy OpenText Documentum CM for SAP Solutions.

Notes

- In most deployments, all of the pods should run with the same time zone. If the time zone of the pods differs, unexpected behavior may result. OpenText recommends that all pods in the deployment run with the same time zone.
- If you specify a pod time zone, the value of `global.timeZone` is not applied to your deployment. The following time zone command arguments should be used together. If you set one time zone using one of the following arguments, be sure to use the remaining arguments to set the other time zones.

Content Server
To set the time zone of the `otcs` (Content Server) pods, use the following Helm command argument

```
--set otcs.config.timeZone=<timeZone>
```

OTDS
To set the time zone of the `otds` (OpenText Directory Services) pod, use the following Helm command argument

```
--set otds.config.timeZone=<timeZone>
```

6.7 Content Server

This section describes changes that you can make to the deployment of Content Server.

6.7.1 Content Server Administration Settings File

A Content Server Administration Settings file is a collection of administration settings that have been saved from a Content Server deployment. When you import that Administration Settings file to a different Content Server deployment, the saved administration settings are applied to it.

Tip: For information on exporting and importing Content Server Administration Settings file, see *OpenText Content Management Admin Help - System Administration (LLESWBA-H-AGD)*.

You can use an Administration Settings file in two different ways:

- Add the Administration Settings file to the `otxecm` Helm chart.
- Create a ConfigMap to hold the Administration Settings file.

6.7.1.1 Add the Administration Settings file to the otxecm Helm chart

To include your own Administration Settings file in a cloud deployment of OpenText Documentum CM for SAP Solutions, enable the loading of an administration settings file and include it in the `otxecm` Helm chart.

Enable the use of an administration settings file
Include the Helm command argument `--set otcs.loadAdminSettings.enabled=true`

Add the administration settings file to the Helm chart
Place an Administration Settings file named `adminSettings.xml` in the `../otxecm/charts/otcs` folder.

The configuration saved in your `adminSettings.xml` file will be applied when you deploy OpenText Documentum CM for SAP Solutions.

6.7.1.2 Create a ConfigMap to hold the Administration Settings file

When you use a ConfigMap to hold the administration settings file, you can create two ConfigMaps: one that is applied on initial deployment and one that is applied whenever a Content Server pod is restarted.

To create the ConfigMaps for the initial and restart administration settings files, you can use the following command:

```
kubectl create configmap <Adminsettings_ConfigMap_Name> --from-file=<Path_To_Adminsettings_Xml_File>
```

Refer to the ConfigMaps using the following Helm command arguments:

Enable the use of an administration settings file
Include the Helm command argument `--set otcs.loadAdminSettings.enabled=true`

Specify the administration settings ConfigMap to apply on initial deployment
`--set otcs.loadAdminSettings.initialConfigmap=<ConfigMap_Name>`

Specify the administration settings ConfigMap to apply on restart
`--set otcs.loadAdminSettings.recurrentConfigmap=<ConfigMap_Name>`

6.7.2 Content Server Applications

Content Server Applications are a special type of application that are built using OpenText WebReports. Some Content Server Applications are present by default in Content Server. WebReports also enables third parties to create their own Content Server Applications. You can use the following Helm command arguments to install or upgrade the Content Server Applications that are present by default in Content Server.

Install
To install every Content Server Application that is present by default on the **Applications Management** administration page, use the following command argument

`--set otcs.config.defaultAppsInstall=true`.

Upgrade
To upgrade every Content Server Application that is present by default on the **Applications Management** administration page, use the following command argument

`--set otcs.config.defaultAppsUpgrade=true`.

Tip: You can also use Init containers to install or upgrade custom Content Server Applications. See "Create Init containers that can be added to the OpenText Documentum CM for SAP Solutions container at deployment time" on page 63.

6.7.3 Content Server File Store

A default deployment of OpenText Content Management deploys an instance of Archive Center as a storage provider for Content Server.

To enable the use of a different storage provider, use any of the following Helm command arguments that you require.

Disable the use of Archive Center

`--set global.otac.enabled=false`

The default value is `true`.

Specify the Storage Provider

`--set otcs.config.documentStorage.type=`*`<Storage_Provider_Type>`*

<Storage_Provider_Type> can be `aws`, `database`, `efs`, `gcp`, or `otac`. The default value is `otac`. If you set a different value, you must set additional configuration parameters.

aws

To use an Amazon Web Services bucket as a storage provider, set the value of `otcs.config.documentStorage.type` to `aws` and include the following Helm command arguments:

AWS Region

`--set otcs.config.awsStorageProvider.region=`*`<Aws_Region>`*

For example: `--set otcs.config.awsStorageProvider.region=ap-south-1`

AWS Bucket Name

`--set otcs.config.awsStorageProvider.bucketName=`*`<Aws_Bucket>`*

For example: `--set otcs.config.awsStorageProvider.bucketName=11-end-bucket`

AWS Secret Key

`--set otcs.passwords.awsSecretKey=`*`<Aws_Secret_Key>`*

Note: You can specify the AWS Secret Key and AWS Access ID in a Secret instead of in a Helm command argument or the `values.yaml` file. For more information, see "Kubernetes Secrets for Passwords and Sensitive Data" on page 45.

AWS Access ID

`--set otcs.passwords.awsAccessId=`*`<Aws_Access_Id>`*

database

To use database storage, configure a database server following the instructions at "Content Server Database Server" on page 66.

efs

To use an External File Store (EFS), set the value of `otcs.config.documentStorage.type` to `efs` and include the following Helm command arguments:.

Path

`--set otcs.config.documentStorage.efsPath=<Efs_File_Path>`

Size

`--set otcs.config.documentStorage.efsStorage=<Storage_Size>`

The default value of <*Storage_Size*> is `1Gi`.

Storage Class

`--set otcs.config.documentStorage.efsStorageClassName=<Storage_Class_Name>`

gcp

To use a GCP bucket as a storage provider, set the value of `otcs.config.documentStorage.type` to `gcp` and include the following Helm command arguments:

GCP Service Account JSON

`--set otcs.config.gcpStorageProvider.serviceAccountJson=<Service_Account_Json_File>`

Add the <*Service_Account_Json_File*> to the root of your `otxecm` Helm chart before your execute the deployment command.

Note: You can specify this value in a Secret instead of in a Helm command argument or the `values.yaml` file. If you do, you do not need to include the JSON file in your Helm chart. For more information, see "Kubernetes Secrets for Passwords and Sensitive Data" on page 45.

GCP Bucket Name

`--set otcs.config.gcpStorageProvider.bucketName=<Gcp_Bucket_Name>`

6.7.4 Content Server Information Protection Administration

OpenText™ Information Protection Administration (IPA) is a Content Server module that integrates with Microsoft Azure Information Protection. Based on business rules, it removes AIP Sensitivity labels when it ingests content, and re-applies them to content when it is retrieved. IPA protects information and prevents data leakage while maintaining your important business processes.

Note: For information on installing and configuring Information Protection Administration, see *OpenText Content Management - Information Protection Administration Guide (LLESCPM-AGD)*.

Information Protection Administration requires a Protected Storage Provider. For instructions on enabling and configuring a Protected Storage Provider, see "Protected Storage Provider and Storage Provider Cache" on page 55.

To enable and configure Information Protection Administration in a containerized deployment of OpenText Documentum CM for SAP Solutions, use the following Helm command arguments.

Enable AIP Processing

`--set otcs.config.contentProtection.enabled=true`

Including this Helm argument enables IPA and automatically configures the following settings:

- The **Enable AIP** check box is selected on the **Enable AIP Processing** administration page.
- The MIP SDK is included with the OpenText Content Management base image at `/opt/mip_sdk/`.
- The **MIPSDK Path** is set to `/opt/mip_sdk/` on the **Specific** Properties page of each DCS server in your deployment.

6.7.5 Content Server Languages

A default deployment of OpenText Documentum CM for SAP Solutions includes all of the officially supported Content Server language packs. The languages are installed but not enabled. You can enable them on the **Configure Languages** administration page. For more information, see *OpenText Content Management Admin Help - System Administration (LLESWBA-H-AGD)*.

6.7.6 Content Server or OpenText Content Management License File

Your deployment of OpenText Documentum CM for SAP Solutions requires a license for Content Server.

You can apply your license by adding it to the `otxecm` Helm chart before deployment, by creating a Secret to store your license, or by applying your license after you have deployed OpenText Documentum CM for SAP Solutions.

6.7.6.1 Add your license to the Helm chart

Before you deploy your product, place your organization’s license file in the `otxecm/charts/otcs` directory. Refer to the license file using Helm command arguments when you run your deployment command.

To include your organization’s Content Server license file:

1. Before you run your Helm deployment command, copy your Content Server license file to the `otxecm/charts/otcs` directory.
2. When you run your Helm deployment command, include the following command arguments:

   Include a license file
   `--set otcs.loadLicense.enabled=true`

   Specify the name of the provided license file
   `--set otcs.loadLicense.filename=<myLicenseFile>`

   Specify the type of license file
   To use an OpenText Content Management license file, add the `--set otcs.config.useExtendedECMLicense=true` command argument.

   To use a Content Server license file, add the `--set otcs.config.useExtendedECMLicense=false` command argument. (You can also simply not include this command argument as its default value is `false`.)

6.7.6.2 Create a Secret to hold the license file

To use a Secret to hold your Content Server or OpenText Content Management license file, create a Kubernetes Secret and refer to it using a Helm command argument:

To create the Secret for your license file, you can use the following command:

```
kubectl create secret generic <License_Secret_Name> --from-file=<Path_To_
License_File>
```

Refer to the license Secret using the following Helm command arguments:

Include a license file
`--set otcs.loadLicense.enabled=true`

Enable the use of a global license Secret
`--set global.existingLicenseSecret=<OTCS_License_Secret_Name>`

6.7.7 Content Server Logon Page Display Name

A default deployment of OpenText Documentum CM for SAP Solutions displays **OpenText Content Management CE <##/#>** on the logon page.

To modify the logon page display name in the deployment, include this parameter in the `helm install` command:

```
--set otcs.config.otds.displayName='<custom_string>'
```

6.7.8 Content Server Log Files

Use the Helm command arguments in this topic to enable the production of the following Content Server log files:

Security Logs

`--set otcs.config.enableSecurityLogs=true`

For information on Content Server logs, see *OpenText Content Management Admin Help - System Administration (LLESWBA-H-AGD)*.

System Monitoring Logs

`--set otcs.config.enableSysmonLogs=true`

For information on Content Server System Monitoring, see *OpenText Content Management Admin Help - System Administration (LLESWBA-H-AGD)*.

6.7.9 Content Server Object Importer

OpenText Object Importer enables the automatic import of any number of items from the file system into Content Server. You can enable the configuration of Object Importer by using the `--set otcs.objectimporter.enabled=true` Helm command argument.

When `otcs.objectimporter.enabled` is set to `true`, a dedicated PVC for Object Importer is created, which includes the following directories:

Import Directory

The `/opt/opentext/sftp/oi/data` directory is created as the default **Import** folder.

Control File

The `/opt/opentext/sftp/oi/control` directory is created. On the Content Server **Configure Object Importer** administration page, this directory is set as the **Control File Directory Path**.

Working Directory

The `/opt/opentext/sftp/oi/work` directory is created. On the Content Server **Configure Object Importer** administration page, this directory is set as the **Working Directory Path**.

Log Directory
: The `/opt/opentext/sftp/oi/logs` directory is created. On the Content Server **Configure Object Importer** administration page, this directory is set as the **Log Directory Path**.

Once you have deployed OpenText Documentum CM for SAP Solutions, you can use these settings to upload Documents and other items to Content Server. You will need to place the documents to be uploaded in the `/opt/opentext/sftp/oi/data` directory, create an Object Importer Control File, and run the import. For more information, refer to the help that is available on the Content Server **Configure Object Importer** administration page.

6.7.10 Content Server Port

In a default deployment of OpenText Documentum CM for SAP Solutions, the `otcs` (Content Server) service runs on port 80.

To set the port of the `otcs` service to a number other than 80, use the following Helm command argument:

`--set otcs.config.port=`*`<Port_Number>`*.

Important

The `otcs` port can be modified only on the initial deployment. Do not change the `otcs` port after you have deployed OpenText Documentum CM for SAP Solutions.

6.7.11 Protected Storage Provider and Storage Provider Cache

A Protected Storage Provider is required for Information Protection Administration and the Advanced Media Add-on. The Advanced Media Add-on also requires a Storage Provider Cache.

- For information on Information Protection Administration, see "Content Server Information Protection Administration" on page 51.
- For information on the Advanced Media Add-on, see "OpenText Content Management, Add-on Advanced Media Viewer" on page 61.

To enable the creation of a Protected Storage Provider and Storage Provider Cache, use the following Helm command arguments.

Protected Storage Provider

Protected Storage Provider Path
: `--set otcs.config.contentProtection.path='/`*`<directory>`*`'`

This command argument creates a Protected Storage Provider in a subdirectory of the `/opt/opentext/` directory. For example, `--set otcs.config.contentProtection.path='/ipa'` creates a Protected Storage Provider at `/opt/opentext/ipa/`. Upon deployment, this **Provider Path** appears on the **Configure Protected Storage Provider** administration page.

Protected Storage Provider Size

`--set otcs.config.contentProtection.storage=<size>`

This command argument sets the size of the Protected Storage Provider to <*size*>. For example, `--set otcs.config.contentProtection.storage=1Gi` allocates 1 Gi to the Protected Storage Provider.

Storage Provider Cache

Enable a Storage Provider Cache

`--set otcs.config.storageProviderCache.enabled=true`

The default value is `false`.

Set the size of the Storage Provider Cache

`--set otcs.config.storageProviderCache.storage=<Size>`

Enter the value of <*Size*> as a number of bytes. For example, enter `500 GB`, `1 TB`, `1.5 TB` or `2 TB`. (These are the values that can be selected on the **Configure Storage Provider Cache Path** administration page in OpenText Content Management.)

Set the name of the Storage Provider Cache

`--set otcs.config.storageProviderCache.dmtshost='<Host_Name>'`

If you are configuring a Storage Provider Cache for use with the Advanced Media Add-on, provide the following Host Name as a value of the above command argument: `dmts-service`.

Set the type of storage used for the Storage Provider Cache

`--set otcs.config.storageProviderCache.storageClassName=<Storage_Class_Type>`

For the value of `storageClassName`, enter a storage volume type. If you are using a Storage Provider Cache for the Advanced Media Add-on, the value of `storageClassName` must be `RWX`.

6.7.12 Content Server Syndication

Content Server Syndication is a distributed solution for remote sites that improves the experience of Content Server for users working at those remote sites.

Caution

The implementation of Content Server Syndication has specific conditions and requirements. Be sure to review the relevant documentation before you enable Content Server Syndication, for example: *OpenText Content Management - Syndication Administration Guide (LLESQDS-AGD)*.

A Content Server Syndication environment consists of a primary server and one or more remote servers. To enable Content Server Syndication in a containerized deployment of OpenText Documentum CM for SAP Solutions, you perform the following steps in order. Each one is explained in further detail below.

1. Deploy the primary Syndication server.
2. On the primary Syndication server, configure Syndication and register a remote Syndication server.
3. Deploy the remote Syndication server.
4. Configure Syndication on the remote Syndication server.

6.7.12.1 Deploy the Primary Syndication Server

To deploy Content Server as a primary Syndication server, include the following command arguments:

Enable Syndication
: Include the command argument: `--set otcs.config.syndication.enabled=true`. This is equivalent to enabling Syndication on the Content Server **Configure Server Parameters** administration page.

Make the current deployment the Primary Syndication Server
: Include the command argument `--set otcs.config.syndication.isPrimary=true`. For the primary Syndication server, the value of this setting must be `true`.

Set the site ID of the Primary Syndication Server
: Include the command argument: `--set otcs.config.syndication.siteid='0'`. For the primary Syndication server, this value must be `'0'`.

Set the Site Name of the Primary Syndication Server
: Include the command argument: `--set otcs.config.syndication.sitename='<sitename>'`. This is equivalent to entering <*sitename*> as the value of **Site Name** on the Content Server **Configure Server Parameters** administration page.

Set the Local Base URL and port
: The **Local Base URL and Port** is a setting on the **Configure Syndication Site Parameters** page. In a Helm deployment of OpenText Documentum CM for SAP

Solutions, this setting is a combination of the two following values: of `qdsUrl` and `port`.

- **qdsUrl**

  Include the command argument: `--set otcs.config.syndication.qdsUrl='<Url>'`. The value of *<Url>* is typically the public URL of your OpenText Documentum CM for SAP Solutions deployment.

- **port**

  Include the command argument: `--set otcs.config.syndication.port=<port>`. The value of *<port>* is the number of a TCP port that is available on your primary Syndication server.

6.7.12.2 Configure the Primary Syndication Server

Once the primary Syndication Content Server is running, the **Configure Server Parameters** admin page shows that Syndication is enabled, the **Site Name** is set to the value of `otcs.config.syndication.sitename`, and there is no value for **Remote Site ID**.

On the **Configure Syndication Site Parameters** administration page, the **Local Base URL and Port** box contains the values *`<qdsURl>`*: *`<port>`* that you set in the Helm command arguments.

To test the functionality, enter the following information and then click **Test**.

- The service name of the Content Server front end, `otcs-frontend`, in the **Content Server Host Name** box.
- The **Local Content Server Admin Username**.
- The **Local Content Server Admin Password**

Make the following configuration changes in the Content Server Syndication administration pages:

- On the **Configure Ingress URL** administration page, set the value of the **Primary Content Server URL** to *`<qdsUrl>`*. Click **Save Changes**.
- On the **Configure Syndication Service** page, click **Configure Syndication Service** and verify that logon to the QDS service is successful.
- On the **Manage Syndication Sites** administration page, register a **Remote Syndication Site** and note the following information. You will use these values when you deploy and configure a remote Syndication server.

  Site ID
  The **Site ID** is automatically generated and cannot be edited.

  Site Name
  The **Site Name** is a plain-language name that you assign to your remote Syndication server.

Pass Key
: The **Pass Key** is a case-sensitive string of letters, numbers, and symbols that the remote Syndication server will use to access the primary Syndication server.

6.7.12.3 Deploy the Remote Syndication Server

To deploy Content Server as a remote Syndication server, include the following command arguments:

Enable Syndication
: Include the command argument: `--set otcs.config.syndication.enabled=true`. This is equivalent to enabling Syndication on the Content Server **Configure Server Parameters** administration page.

Make the current deployment a Remote Syndication Server
: Include the command argument `--set otcs.config.syndication.isPrimary=false`. For a remote Syndication server, the value of this setting must be `false`.

Set the site ID of the Remote Syndication Server
: Include the command argument: `--set otcs.config.syndication.siteid=<site_ID>`. Set the value of <*site_ID*> to the value of the **Site ID** that you generated on the **Add Syndication Site** administration page on the primary Syndication server.

Set the Site Name of the Remote Syndication Server
: Include the command argument: `--set otcs.config.syndication.sitename='<sitename>'`. Set the value of <*sitename*> to the value that you entered for **Site Name** on the **Add Syndication Site** administration page on the primary Syndication server.

Set the Primary Base URL and port
: The **Primary Base URL and Port** is a setting on the **Configure Syndication Site Parameters** page. In a Helm deployment of OpenText Documentum CM for SAP Solutions, this setting is a combination of the two following values: of `qdsUrl` and <*port*>.

- **qdsUrl**

  Include the command argument: `--set otcs.config.syndication.qdsUrl=<Url>`. The value of <*Url*> is typically the public URL of your product deployment.

- **port**

  Include the command argument: `--set otcs.config.syndication.port=<port>`'. The value of <*port*> is the number of a TCP port that is available on your remote Syndication server.

6.7.12.4 Configure the Remote Syndication Server

Once the remote Syndication Content Server is running, the **Configure Server Parameters** admin page shows that Syndication is enabled, and the **Site Name** and the **Remote Site ID** boxes contain the values that you set in the remote Syndication server deployment command.

Make the following configuration changes in the Content Server Syndication administration pages.

- On the **Configure Syndication Site Parameters** administration page: .
  - Set the value of the **Local Base URL and Port** to `<qdsUrl>`:`<port>`, where *<qdsUrl>* is the value that you set in your Helm deployment command and *<port>* is the number of a TCP port that is available on the remote Syndication server (the local machine).
  - To test the functionality, enter the following information and then click **Test**.
    - The service name of the Content Server front end, `otcs-frontend`, in the **Content Server Host Name** box.
    - The **Local Content Server Admin Username**.
    - The **Local Content Server Admin Password**
  - Set the value of the **Primary Base URL and Port** to `<qdsUrl>` (with no port specified).
  - In the **Activation Pass Key** box, enter the **Pass Key** that you created when you created a new remote Syndication server on the **Add Syndication Site** administration page on the primary Syndication server.
  - To test the functionality, enter the **Primary Content Server Admin Username** and the **Primary Content Server Admin Password**, and then click **Test**.
  - Once both of the above tests pass successfully, click **Save Changes** at the bottom of the page.
- On the **Configure Ingress URL** administration page, set the value of **Remote Content Server URL** to `<qdsUrl>`, and then click **Save Changes**.
- On the **Configure Syndication Service** page, click **Configure Syndication Service** and verify that logon to the QDS service is successful.

6.7.12.5 Complete the Syndication Setup

Once you have completed the above steps, you will have additional steps to perform. For example:

- Enter the Ingress URL of the remote Syndication server on the primary Syndication server's **Configure Ingress URL** page.
- Import users and groups from OTDS, on the primary Syndication server's **Import Users and Groups** administration page.

Consult the Syndication documentation for information on these and other steps that you may need to perform.

6.7.13 Content Server Threads

A default deployment of OpenText Documentum CM for SAP Solutions configures Content Server nodes to run with 8 threads.

To set the number of Content Server threads that the various Content Server nodes run, use the following Helm command arguments:

Front-end nodes
`--set otcs.contentServerFrontend.threadsNumber=`*`<Number_of_Threads>`*.

Admin server node
`--set otcs.contentServerAdmin.threadsNumber=`*`<Number_of_Threads>`*.

Back-end Search nodes
`--set otcs.contentServerBackendSearch.threadsNumber=`*`<Number_of_Threads>`*

6.7.14 OpenText Content Management, Add-on Advanced Media Viewer

The Advanced Media Add-on enables streaming audio and video in Content Server. To deploy the Advanced Media Add-on, you require an additional Helm chart and docker image. You must also configure Content Server to use Advanced Media Add-on, once you have completed your OpenText Content Management deployment and Content Server is running.

OpenText Content Management, Add-on Advanced Media Viewer requires a Protected Storage Provider and a Storage Provider cache. For instructions on enabling and configuring a Protected Storage Provider and a Storage Provider Cache, see "Protected Storage Provider and Storage Provider Cache" on page 55.

Helm chart and docker image
For information on the Advanced Media Add-on Helm chart and docker image, see *OpenText Content Management with media management - Cloud Deployment Guide (MEDMGTMV-ICD)*.

OpenText Content Management

For information on enabling and configuring the Advanced Media Add-on, refer to the online help that is available for the **Configure Media Conversion Settings** administration page.

Tip: To access the **Configure Media Conversion Settings** administration page, open the OpenText™ Content Management Administration page, and then click **Features**. On the **Configure Features** page, click **Media Conversion Settings**.

6.7.15 Add Modules, Content Server Applications, and Patches

If you require additional Content Server modules or patches with your deployment, you can extend the OpenText Documentum CM for SAP Solutions base image.

Alternatively, you can install Content Server modules, Content Server Applications, and patches using Init containers. Using Init containers to add to the base image can allow for more flexibility in building images for different purposes or clients.

This section explains how to use both methods. Whichever one you choose, you require a Linux host machine that runs Docker and a Docker repository that you can push the images to once they are built.

Bear in mind that when you add modules to the base image, it is up to you to ensure that the modules are suitable for use with Linux and PostgreSQL and that all module prerequisites are satisfied. If a module that you want to install depends on the presence of other modules, you must include those modules. Any module that you install must be a typical module that you install by copying the installation files to the `<Content_Server_home>/staging` directory.

6.7.15.1 Extend the OpenText Documentum CM for SAP Solutions image

Complete the steps in this topic to extend your OpenText Documentum CM for SAP Solutions image. The OpenText Documentum CM for SAP Solutions image must be accessible from your local Docker repository. Place the modules and patches in the specific directories listed below. When all is ready, execute a `docker build` command that references a file named `Dockerfile_extend` to incorporate the modules and patches into your base image.

To add modules and patches to the base image:

1. Obtain the `Dockerfile_extend` file from My Support. Save it to a directory on a Linux host machine that has Docker installed.

2. Below the directory that contains the `Dockerfile_extend` file, create any of the following subdirectories that you require.

   - `./cs_install/modules`
   - `./cs_install/patches`

3. Place the items to be added to your image in the subdirectories, as follows:

   Modules
   Place modules in the `./cs_install/modules` directory. Modules should be added with their `.tar` extension. Do not untar the modules when you add them.

   Patches
   Place any patch files that you want to add to the image in the `./cs_install/patches`.

4. Execute the following `docker build` command:

   `docker build -f Dockerfile_extend . --build-arg base_image=`*`<Base_Image_Name>`*`:`*`<Base_Image_Tag>`*` -t `*`<Extended_Image_Name>`*`:`*`<Extended_Image_Tag>`*

   Example: For example, to extend the `otcs:24.2.0` image and create a new image called `otcs_enhanced:24.2.0`, run the following command:

   `docker build -f Dockerfile_extend . --build-arg base_image=otcs:24.2.0 -t otcs_enhanced:24.2.0`

   In the output of the command, a series of messages indicates that the new image is being built. When the command completes successfully, you see messages indicating:

   ```
   Successfully built <Image_ID>
   Successfully tagged <Image_Name>:<Tag_Name>
   ```

5. Once the extended image is built, push it to your image repository. Then reference the name and tag of the extended image when you execute the Helm command that deploys OpenText Documentum CM for SAP Solutions.

6.7.15.2 Create Init containers that can be added to the OpenText Documentum CM for SAP Solutions container at deployment time

Complete the steps in this topic to build one or more Init containers and add them to your OpenText Documentum CM for SAP Solutions container at deployment time.

To add modules, patches, and Content Server Applications to the base image:

1. Obtain the **Dockerfile_init** `Dockerfile` from My Support. Save it to a directory on a Linux host machine that has Docker installed. You can name the directory whatever you like.

2. Create the following subdirectories below the directory that contains the `Dockerfile`.

   - `./extensions/apps/install/`
   - `./extensions/apps/upgrade/`
   - `./extensions/manifest/`
   - `./extensions/modules/`

- `./extensions/patch/`

3. Build Init containers one at a time. Place a module, patch, or Content Server Application in the appropriate directory. If you have more than one Init container to build, repeat the steps in this procedure as many times as necessary.

   Content Server Application

   - **Install**

     To include a Content Server Application in your OpenText Documentum CM for SAP Solutions deployment and install it using a Helm command argument, place it in the `./extensions/apps/install/` directory.

   - **Upgrade**

     To upgrade an existing Content Server Application in your OpenText Documentum CM for SAP Solutions during the Helm deployment, place the upgrade version of your Content Server Application in the `./extensions/apps/upgrade/` directory.

   Module
   Place a module in the `./extensions/modules/` directory. Add the module with its `.tar` extension. Do not untar the module when you add it.

   Patch
   Place a patch in the `./extensions/patch/` directory.

4. Obtain the latest publicly available Alpine Linux image and store it in the same repository as your OpenText Documentum CM for SAP Solutions image.

5. Execute the following `docker build` command:

   `docker build -f Dockerfile_init . --build-arg base_image_tag=`*`<Alpine_Image_Tag>`*` -tag `*`<Init_Image_Name>`*`:`*`<Init_Image_Tag>`*

   Example: For example, to build an Init container image called `myModule:latest` using `Alpine:3.15.4` as the base image, run the following command:

   ```
   docker build -f Dockerfile_init . --build-arg base_image_tag=3.15.4
    --tag myModule:latest
   ```

   In the output of the command, a series of messages indicates that the new image is being built. When the command completes successfully, you see messages indicating:

   ```
   Successfully built <Image_ID>
   Successfully tagged <Image_Name>:<Tag_Name>
   ```

6. Once the Init image is built, push it to your image repository. Then, include the following Helm command argument when you deploy OpenText Documentum CM for SAP Solutions: `--set otcs.config.extensions.enabled=true`.

Specify the following Helm command arguments for each Init container that you are adding to the deployment. (If you are using a manifest container, include these Helm command arguments for it as well.) Increment the number in square brackets for each Init container that you add.

- `--set otcs.initContainers[0].name=`*`<Init_Container_Image_Name>`* This name can be whatever you want.
- `--set otcs.initContainers[0].image.source=`*`<Image_Source>`*. The registry that contains your images. Include this parameter, even if it's the same image source as the one specified by `global.imageSource`.
- `--set otcs.initContainers[0].image.name=`*`<Image_Name>`* This is the name that you set in your `docker build` command.
- `--set otcs.initContainers[0].image.tag=`*`<Image_Tag>`* This is the tag that you set in your `docker build` command.

6.8 Database Servers

Follow the instructions in this section to configure OpenText Content Management to connect to the database servers that are required for their operation.

When you deploy OpenText Content Management, you can choose to have Helm create the databases for you, or you can connect to an existing database.

Note: The use of a containerized PostgreSQL database server is suitable for a development environment, but OpenText recommends that you do not use one for a production environment.

An existing database can be empty (in which case Helm creates the required schema) or pre-populated, in which case Helm connects to the database but does not create any tables.

Important

To use Helm to connect to an *existing* database, create the database before you run your Helm deployment. When you create the database, be sure to create it with all required components. For example, a PostgreSQL database created for Content Server or OTDS requires the `pg_trgm` PostgreSQL extension. You can verify the database requirements for a given product in its product installation guide or in its release notes.

6.8.1 Content Server Database Server

You can connect a Microsoft® SQL Server®, Oracle or PostgreSQL database server to house your Content Server database.

6.8.1.1 SQL Server for Content Server

To use a SQL Server database server to house your Content Server database, include any of the command arguments listed in this section that you require for your deployment.

6.8.1.1.1 Use an Existing Database

Use this argument to specify that the database exists already, either as an empty database in which Helm can create the tables, or as a fully populated Content Server database. If you want Content Server to create the database, keep the default value for this setting: `false`.

```
--set otcs.config.database.useExistingDatabase=true
```

6.8.1.1.2 Connect to the Database Server

Specify SQL Server as the database server type, and provide the host name and port of the database server.

Specify the database server type

```
--set otcs.config.database.type=mssql
```

Specify the database server name

```
--set otcs.config.database.hostname=<Database_Host_DNS_Name>
```

Specify the database server port

```
--set otcs.config.database.port=<Content_Server_Database_Port>
```

Note: If it is necessary to specify that the database port is `null`, set the value of the above command argument to `null` and include the following global setting in your deployment command: `--set global.database.port=""`.

6.8.1.1.3 Connect to the Master Database

To create the Content Server database, Content Server requires access to the Master database.

Set the name of the SQL Server administrative database

```
--set otcs.config.database.adminDatabase=<SQL_Server_Admin_DB>
```

The name of the SQL Server Admin database is `master`.

Set the name of the SQL Server administrative user

```
--set otcs.config.database.adminUsername=<DB_Admin_User_Name>
```

Set the password of the SQL Server administrative user

`--set otcs.passwords.database.adminPassword=<DB_Admin_Password>`

This argument is required if the password of the admin user for the Content Server database server has not been set by any other means. See “Passwords and sensitive data” on page 42.

6.8.1.1.4 Create the Content Server Database

Set the names of the Content Server Database and Database User

`--set otcs.config.database.name=<Content_Server_Database_Name>`

`--set otcs.config.database.username=<Content_Server_Database_User_Name>`

Set the initial size and location of the SQL Server data file, and allow it to be extended automatically

Data File location

`--set otcs.config.database.mssql.dbDataFileSpec=<Data_File_Location>`

Data File Size

`otcs.config.database.mssql.dbDataFileSize=<#_of_Megabytes>`.

The default is `500`.

AutoExtend Data File

`--set otcs.config.database.autoExtendDataFile=[true|false]`.

The default value of this setting is `true`. You do not need to set it unless you want it to disable autoextension of the data file.

Set the initial size and location of the SQL Server log file, and allow it to be extended automatically

Log File Location

`--set otcs.config.database.mssql.dbLogFileSpec=<Log_File_Location>`

Log File Size

`--set otcs.config.database.mssql.dbLogFileSize=<#_of_Megabytes>`.

The default is `500`.

Autoextend Log File

`--set otcs.config.database.autoExtendLogFile=[true|false]`.

The default value of this setting is `true`. You do not need to set it unless you want to disable autoextension of the log file.

6.8.1.2 PostgreSQL for Content Server

To use a PostgreSQL database server to house your Content Server database, include any of the following command arguments that you require for your deployment.

If you create the Content Server database and database user prior to deployment (rather than letting Helm create the database for you), ensure that your PostgreSQL database has the `pg_trgm`, `pgcrypto`, and `unaccent` extensions. Also, ensure that the Content Server database user has the `superuser` privilege. For more information, see *OpenText Content Management - Installation Guide (LLESCOR-IGD)*.

Use the below Helm command arguments to specify the name of your Content Server database and database user.

To specify that the database exists already

`--set otcs.config.database.useExistingDatabase=true`

Use this argument to specify that the database exists already, either as an empty database in which Helm can create the tables, or as a fully populated Content Server database.

To connect to the Content Server database server

`--set otcs.config.database.hostname=<Database_Host_DNS_Name>`

`--set otcs.config.database.port=<Content_Server_Database_Port>`

`--set otcs.config.database.name=<Content_Server_Database_Name>`

To set the name of the admin user for the Content Server database server

`--set otcs.config.database.adminUsername=<DB_Server_Admin>`

This argument is required if the admin user’s name is not `postgres`.

To set the password of the admin user for the Content Server database server

`--set otcs.passwords.database.adminPassword=<DB_Admin_Password>`

This argument is required if the password of the admin user for the Content Server database server has not been set by any other means. See “Passwords and sensitive data” on page 42.

To set the name of the default database on the Content Server database server

`--set otcs.config.database.adminDatabase=`*<Default_Database_Name>*

This argument is required if the default database’s name is not `postgres`.

To set the name of the Content Server database user

`--set otcs.config.database.username=<Content_Server_Database_User>`

This argument is required if the name of the Content Server database user is not `cs`

To set the password of the Content Server database user

`--set otcs.passwords.database.password=<Password>`

This argument is required if the Content Server database user’s password has not been set by any other means. See “Passwords and sensitive data” on page 42.

6.9 OTDS

This section describes changes that you can make to the deployment of OTDS.

6.9.1 Use a different OTDS instance

A default deployment of OpenText Documentum CM for SAP Solutions deploys an OTDS instance in your Kubernetes cluster that provides authentication services to the other OpenText components in the deployment.

To use a different OTDS server that is located outside of the `otxecm` Kubernetes cluster, such as your organization’s OTDS server:

- Include the following Helm command argument to prevent OTDS from being deployed as a container: `--set global.otds.enabled=false`
- Configure Archive Center, Content Server, Intelligent Viewing, and OpenText Content Management – Document Generation, as required, to connect to an OTDS server that is external to the cluster.

Content Server

To use a different OTDS with Content Server, you may need to modify the OTDS URLs and specify a different OTDS Resource that Content Server will use.

OTDS URLs

By default, Content Server connects to OTDS at the address that you specify as the `otdsPublicUrl`. If your deployment requires modification of the **OTDS Server URL** or the **OTDS Sign In URL**, include one or both of the following Helm command arguments:

Content Server: OTDS Server URL

If Content Server needs to connect to OTDS using an address that is different from the `otdsPublicUrl`, set the address explicitly using the `otcs.config.otds.serverUrl` property.

```
--set otcs.config.otds.serverUrl=<OTDS_URL>
```

Include the OTDS port in the URL if OTDS is not running on port 80.

Example: `--set otcs.config.otds.serverUrl=https://otds.example.com:8080`

Tip: This value appears as the value of **OTDS Server URL** in the **Global Integration Settings** area of the **Directory Services Integration Administration** page in Content Server. It is the URL that the containers in the deployment use to connect to OTDS. .

Content Server: OTDS Sign In URL

If you need Content Server to use an **OTDS Sign In URL** that is different from the `otdsPublicUrl`, set the address explicitly using the `otcs.config.otds.signInUrl` property.

Include the OTDS port in the URL if OTDS is not running on port 80.

`--set otcs.config.otds.signInUrl=<OTDS_Sign_in_URL>`

Example: `--set otcs.config.otds.signInUrl=https://otds.example.com:8080`

Tip: This value appears as the value of **OTDS Sign In URL** in the **Local Integration Settings** area of the **Directory Services Integration Administration** page in Content Server. It is the URL that allows users to connect to OTDS over the Internet.

OTDS Resource for Content Server

By default, the `otcs` Helm chart specifies that the name of the OTDS Resource for Content Server is `cs`. You can change this, if you wish. If you connect to an existing OTDS that already has a Resource named `cs`, you may be required to change this value to avoid a name collision.

To use an OTDS Resource for Content Server that is not named `cs`, use the following command: `--set otcs.config.csResourceName="<Resource_Name>"`

If you include this command, an OTDS Resource named <*Resource_Name*> will be created for Content Server and an Access Role named **Access to <*Resource_Name*>** will also be created.

6.9.2 OTDS port

In a default deployment of OpenText Documentum CM for SAP Solutions, the `otds` service runs on port 80.

To set the port of the `otds` service to a number other than 80, use the following Helm command arguments in your Helm deployment command. Set <*Port_Number*> to the same number in each argument.

Configure OTDS to run on a specific port

`--set otds.port=<Port_Number>`.

Configure Content Server to connect to OTDS on a specific port

`--set otcs.config.otds.port=<Port_Number>`

Important

The `otds` port can be modified only on the initial deployment. Do not change the `otds` port after you have deployed OpenText Content Management.

6.9.3 Set additional Trusted Sites

The OTDS **Trusted Sites** page lists addresses that Directory Services trusts and allows to provide a forwarding address. During authentication, if a referring URL contains a forwarding address, Directory Services will redirect the user's browser to that address, if the referring URL is trusted.

To add to the list of URLs that OTDS trusts, use the following Helm command argument:

```
--set otcs.config.otds.trustedSites[#]='<URL>'
```

where <#> is a sequence number that starts with zero and <*URL*> is a value permitted by OTDS on the **Trusted Sites** page.

Example 6-1: To Add Trusted Sites

To add `www.example1.com` and `www.example2.net` to the list of Trusted Sites in OTDS, include the following command arguments in your Helm deployment command:

```
--set otcs.config.otds.trustedSites[0]='www.example1.com'

--set otcs.config.otds.trustedSites[1]='www.example2.net''
```

For more information on Trusted Sites in OTDS, refer to the help available on the OTDS **Trusted Sites** page.

Chapter 7

Verify the Deployment

7.1 Content Server

Two types of Content Server pods are deployed.

Front-end Server
One or more front-end Content Server instances are deployed, with the name **otcs-frontend**. To verify the operation of a front-end server, log on to it as a Content Server user. Construct the URL of the Content Server front-end server, using the DNS name that you specified in your Helm deployment command, as follows:

`https://<DNS_name>/cs/cs/app`

For example: `https://otcs.csp.example.com/cs/cs/app`

When the Content Server logon page appears, log on to Content Server as `Admin`.

Admin server
One Content Server Admin server is deployed, with the name **otcs-admin**. To verify that it is operating, connect to a Content Server front-end workload, and verify in the user interface that the Admin server is functioning correctly.

7.1.1 Content Server Deployment Log File

During the installation of OpenText Documentum CM for SAP Solutions, the progress of the Content Server deployment is written to the `/opt/opentext/cs_persist/contentserver.log` (`$AUTOMATION_LOG`) file. You can review the instance's `$AUTOMATION_LOG` file to verify that the deployment was completely successful or to investigate the cause of a failed deployment.

To review the contents of the `$AUTOMATION_LOG` file, use the `kubectl logs` command. To read the `$AUTOMATION_LOG` file of a Content Server pod as it is being deployed, run `kubectl logs --follow` *`<Content_Server_Pod_Name>`*. For example, run `kubectl logs --follow otcs-admin-0`.

You can also view the `$AUTOMATION_LOG` file by logging onto a pod and using a text editor to view the file, as follows:

To review the Content Server deployment logs:

1. Display the current running pods, using `kubectl get pods`. Obtain the name of the pod whose `$AUTOMATION_LOG` file you wish to view.

2. Use the `kubectl logs` command to view the pods's logs. For example, to monitor the log output of the `otcs-admin-0` pod, run `kubectl logs otcs-admin-0 --follow`.

Chapter 8

Scale the Deployment

In a default deployment of OpenText Documentum CM for SAP Solutions, the `otcs-frontend` stateful set is deployed with a single replica and the `otcs-backend-search` stateful set is deployed with zero replicas.

You can scale your deployment of OpenText Documentum CM for SAP Solutions by modifying the number of replicas in the `otcs-frontend` stateful set. To scale an application, you increase or decrease the number of replicas.

To set the number of replicas in either of these stateful sets, use a `helm upgrade` command with the appropriate command argument:

Front-end Content Server instance

`--set otcs.contentServerFrontend.replicas=<#>`

Note: You can also scale the deployment using `kubectl` as follows:

```
kubectl scale sts <Content Server instance> --replicas=<Number_Of_Replicas>.
```

However, Helm is not aware of `kubectl scale` commands, so you must repeat this command each time you use Helm to modify your deployment.

Chapter 9

Upgrade the Deployment

In this chapter, you will find basic upgrade instructions, a description of how to perform upgrade logging, and a description of any specific steps you may need to take when you upgrade from a particular version of OpenText Documentum CM for SAP Solutions. This guide presumes that you are upgrading from the current or previous major version. If you are upgrading from an earlier major version, refer to an earlier version of this guide for specific instructions that pertain to the version that you are upgrading from.

9.1 Upgrade OpenText Documentum CM for SAP Solutions

To upgrade your OpenText Documentum CM for SAP Solutions deployment, run a `helm upgrade` command with similar parameters to the `helm install` that you used to deploy it.

For example, if you deployed OpenText Documentum CM for SAP Solutions using the sample command provided in "Deploy OpenText Documentum CM for SAP Solutions Containers" on page 32, you would run the same command but start it with `Helm upgrade`, not `Helm install`.

Tip: For a list of your running Helm releases, run: `helm ls`.

9.2 Upgrade Logging

Container logging during an upgrade is done the same way as during an initial deployment. See "Container Deployment Logging" on page 38.

To obtain logging for each specific application in the deployment, set whichever logging parameters you require in the application. For example, to increase Content Server log levels during an upgrade, use the Content Server **Configure Instance Log Settings** or **Temporary Log Settings** administration page.

Tip: If you set `otcs.containerLogLevel=DEBUG`, Content Server Thread and Connect logs are automatically set to level 2 (INFO). They remain at this level after deployment, until you reset them. See "Application Logging" on page 39.

9.3 Upgrade OpenText Documentum CM for SAP Solutions 23.2 or earlier

In OpenText Documentum CM for SAP Solutions 23.3 and later, there is no longer support for the PostgreSQL 11 database server. A PostgreSQL 13 database is required. To manage the change from PostgreSQL 11 to PostgreSQL 13, perform the following steps before you run your `Helm upgrade` command.

The following instructions presume that you have both your existing PostgreSQL 11 database server and your new PostgreSQL 13 database server running. To ensure database consistency, prevent users from accessing your deployment of OpenText Documentum CM for SAP Solutions while you perform the following steps.

Tip: You do not need to perform steps 2 and 3 if your new PostgreSQL 13 database server has the same name as the PostgreSQL 11 database server that is no longer being used for databases in your deployment.

To transition from PostgreSQL 11 to PostgreSQL 13:

1. Back up (export) each OpenText Documentum CM for SAP Solutions database in your PostgreSQL 11 database server.
2. Restore (import) each database from step 1 to your PostgreSQL 13 database server. In the PostgreSQL 13 database server, ensure that the databases and tables are owned by the by the same users, and that all database names, user names and passwords are the same as in the PostgreSQL 11 database server.
3. In the OTXECM 23.3 or later Helm chart, update your chart values to reflect the host name of the new PostgreSQL 13 database server.
4. Proceed with your `Helm upgrade` command to upgrade your deployment of OpenText Documentum CM for SAP Solutions 23.2 or earlier.

Chapter 10

Adapt the installed system

10.1 Proxy communication of OpenText Documentum CM and OpenText Content Management

If your OpenText Content Management server installation uses an outbound communication to OpenText Documentum CM that is restricted to proxy, you have to configure the proxy communication in your `opentext.ini`.

This setup is configured by your service representative if you bought OpenText Documentum CM for SAP Solutions in the OpenText Content Management system.

To configure proxy communication:

1. Open the `opentext.ini`.
2. Adapt the `Java VMOptions` and add the host and port of your OpenText Content Management system.

   Example: `JavaVMOption_5=-Dhttps.proxyHost=10.96.196.146`

   `JavaVMOption_6=-Dhttps.proxyPort=8888`

10.2 Disable full-text search

The OpenText Documentum CM for SAP Solutions solution keeps all files in your OpenText Documentum CM system. A full-text search in OpenText Content Management is not possible. You can save resources if you disable the feature.

To disable the full-text search:

1. Log into Content Server of the Core system.
2. Click **OpenText Content Management Administration > Search Administration > Open the System Object Volume**.
3. Open the **Enterprise Data Source Folder** context menu and click **Delete**.

Chapter 11

Appendix A - Glossary of Terms

Here is a brief explanation of some of the terms used in this document.

Container
A Docker container is an instance of a Docker image.

Cluster
A Kubernetes Cluster is composed of multiple Kubernetes Nodes. The Cluster intelligently handles distributing work to the individual nodes

Helm Chart
A Helm chart is a collection of files that describe a related set of Kubernetes resources. The name of the chart is the directory within which the files are stored.

Image
A container image is built up from a series of layers. Each layer represents an instruction in the image's `Dockerfile`. Each layer except the last one is read-only. An image name is made up of slash-separated name components, optionally prefixed by a registry host name

Node
A node is the smallest unit of computing hardware in Kubernetes. It is a representation of a single machine in your cluster.

Pod
A Kubernetes pod is a group of one or more containers that are deployed together on the same host. Containers in a pod share resources and the local network.

Registry
Registries store collections of Repositories. Docker Hub is an example of a registry.

Repository
A repository is a location where container images are stored. A repository can exist on a local computer or on an internet-hosted service, such as GCP. You can push container images to a repository, Docker Hub for example, using the `docker push` command. A single repository can hold many container images.

Tag
Image Tags identify variants of container images. A single image can be given one or more tags. The combination of `repository:tag` can be used to identify the intended location of an image.

Chapter 12

Appendix B - Helm Command Reference

The following table summarizes a number of important Helm `--set` command-line parameters. You can find additional parameters in the `otxecm/values.yaml` file.

Command line parameter	Value or effect of the parameter	Example values
`--set otcs.image.name=`	The name of the Content Suite Platform and OpenText Content Management container image	otcs, otxecm, otxecm-sap-o365-sfdc
`--set otcs.image.tag=`	The container image tag (version) of the Content Server image	16.2.10, 16.2.11, 20.2.0, my-custom-tag
`--set otac.image.tag=`	The container image tag (version) of the Archive Centerimage	16.2.2, 20.2.0
`--set otds.image.tag=`	The container image tag (version) of the OTDS image	16.6.3, 20.2.1
`--set otac.enabled=`	Defines if Archive Center is deployed as container or not	`false` or `true`
`--set otcs.contentStore=`	Defines where content is stored (needed if `otac.enabled=false`)	`database`, `archive`, or `efs`
`--set otcs.database.host=`	Host name of the database server (if it is outside the cluster)	IP address or fully qualified domain name
`--set otcs-db.enabled=`	Defines if Content Server Database gets deployed as container or not	`false` or `true`
`--set otcs.adminPassword=`	The password of the Content Server Admin user	*<Admin_User_Password>*
`--set otcs.contentServerFrontend.replicas=`	Number of Content Server front-end instances to start	1-n
`--set otcs.contentServerFrontend.resources.requests.cpu=`	Number of CPUs to be requested for a Content Server front-end instance	1

Command line parameter	Value or effect of the parameter	Example values
`--set otcs. contentServerFrontend. resources.requests. memory=`	Compute memory to be requested for a Content Server front-end instance	1.5Gi, 8000 Mi
`--set otcs. contentServerFrontend. limits.requests.cpu=`	CPU limit for a Content Server front-end instance	2
`--set otcs. contentServerFrontend. limits.requests.memory=`	Compute memory limit for a Content Server front-end instance	4Gi, 8000 Mi
`--set otcs. contentServerAdmin. resources.requests.cpu=`	Number of CPUs to be requested for a Content ServerAdmin server instance	1
`--set otcs. contentServerAdmin. resources.requests. memory=`	Compute memory to be requested for a Content Server Admin server instance	1.5Gi, 8000 Mi
`--set otcs. contentServerAdmin. limits.requests.cpu=`	CPU limit for Content Server Admin server instance	2
`--set otcs. contentServerAdmin. limits.requests.memory=`	Compute memory limit for Content Server Admin server instance	4Gi, 8000 Mi