opentext™

OpenText™ Documentum™ xPlore

Cloud Deployment Guide

This document provides information about setting up, supporting, installing and configuring, and deploying and configuring, Documentum xPlore on different cloud platforms.

EDCSRCCD220100-CGD-EN-01

OpenText™ Documentum™ xPlore
Cloud Deployment Guide
EDCSRCCD220100-CGD-EN-01
Rev.: 2022-Jan-28

This documentation has been created for OpenText™ Documentum™ xPlore CE 22.1.
It is also valid for subsequent software releases unless OpenText has made newer documentation available with the product, on an OpenText website, or by any other means.

Open Text Corporation

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

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

Table of Contents

Chapter 1

Overview

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

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

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

Chapter 2

Deploying xPlore on Docker Compose environment

2.1 Overview

Compose is a tool for defining and running multi-container Docker applications. With Compose, you can use a YAML file to configure the application's services. Then, with a single command, you can create and start all the services from your configuration.

2.2 Prerequisites

Complete the following activities before you deploy xPlore Docker images using Docker Compose:

1. Download and configure the Docker from the Docker website.
2. Download and configure the Docker Compose from the Docker website.
3. Download the Docker image from OpenText Container Registry. Perform the following tasks:

   a. Log in to OpenText Container Registry using the following command format:

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

   When prompted, provide your OpenText My Support login credentials.

   b. Download the Docker image using the following command format:

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

   The following Docker images are available for download:

Image name	Image tag
dctm-server	22.1.0
dctm-server-ubuntu	22.1.0
dctm_xplore_indexserver	22.1.0
dctm_xplore_indexagent	22.1.0
dctm_xplore_cps	22.1.0

   Note: The `dctm-server-ubuntu` Docker image must be used only for deploying and configuring Documentum Server on Docker.

4. Download the Helm charts from OpenText My Support.

5. Download the Docker Compose file from OpenText My Support.

2.3 Deploy xPlore Docker images using Docker Compose

2.3.1 Docker Compose YAML Sample

Use the following sample xPlore docker compose YAML file, to replace images and environment parameters with the actual value and save it with name docker-compose.

This sample contains two instances of CPS. While you can add or remove instances of CPS, you must keep at least one instance.

```
version: '3'
services:
  indexserver:
    image: indexserver  #replace with your actual indexserver image tag here
    hostname: indexserver
    ports:
      - "9300:9300"
    volumes:
      - xplore:/opt/xPlore/rtdata
  indexagent:
    image: indexagent  #replace with your actual indexagent image tag here
    hostname: indexagent
    ports:
      - "9200:9200"
    environment:
      - ess_host=indexserver
      - docbase_name=your_docbase_name  #replace with actual value
      - docbase_user=your_docbase_user_name  #replace with actual value
      - docbase_password=your_docbase_password  #replace with actual value
      - broker_host=your_doc_broker_host_ip_address  #replace with actual value
      - broker_port=your_doc_broker_host_port  #replace with actual value
      - registry_name=your_registry_name  #replace with actual value
      - registry_user=your_registry_user_name  #replace with actual value
      - registry_password=your_registry_password  #replace with actual value
    depends_on:
      - indexserver
    volumes:
      - xplore:/opt/xPlore/rtdata
  cps:
    image: cps  #replace with your actual cps image tag here
    hostname: cps
    environment:
      - ess_host=indexserver
    depends_on:
      - indexserver
    volumes:
      - xplore:/opt/xPlore/rtdata
  cps1:
    image: cps  #replace with your actual cps image tag here
    hostname: cps1
    environment:
      - ess_host=indexserver
    depends_on:
      - indexserver
    volumes:
      - xplore:/opt/xPlore/rtdata
volumes:
  xplore:
```

2.3.2 Start and stop Docker Compose

2.3.2.1 Start Docker Compose

Change the directory location to the folder where you have saved the docker compose file, and run the following command:

```
docker-compose up
```

You can log in the site, http://<docker host machine IP>:9300/dsearchadmin with the default credentials admin/password to check status of the IndexServer and CPS.

You can log in the site, http://<docker host machine IP>:9200/IndexAgent with the default credential docbase_user/docbase_password you set in the IndexAgent environment section of the YAML file.

2.3.2.2 Stop Docker Compose

Run the following command to stop xPlore:

```
docker-compose down
```

Refer the Docker website for more details.

2.3.3 Configure SSL

To configure SSL of xPlore, perform the following before you run the docker compose:

- Change the port setting to *`9302:9302`* in the `indexserver` section.
- Add the `config_ssl=true` environment parameter in the `indexserver` section.
- Change the port setting to 9202:9202 in the `indexagent` section.
- Add the `ess_port=9302`, `ess_protocol=https`, `config_ssl=true` environment parameters in the `indexagent` section.
- Add the `ess_port=9302`, `ess_protocol=https`, `config_ssl=true` environment parameters in the `cps` section.

You can use the following URLs to check the status of xPlore:

- https://<docker host machine IP>:9302/dsearchadmin
- https://<docker host machine IP>:9202/IndexAgent

2.4 Environment parameters

2.4.1 Environment parameters of IndexServer

Parameter Name	Description
PASSWORD	Default password of dtcm-users, dsearch, dsearch xdb, dsearch admin web console, and dsearch admin web console xdb. **Default value**: password **Required**: No
MEM-ARGS	Part of `JAVA_OPTS`, bash script of starting indexserver. -XX:+UseCodeCacheFlushing -XX:ReservedCodeCacheSize=196m -Xms4096m -Xmx4096m -Xss1024k -XX:+UseCompressedOops -XX:+DisableExplicitGC -XX:-ReduceInitialCardMarks -Djava.awt.headless=true **Default value**: copy default value to modify **Required**: No
CONFIG_SSL	To set the SSL of the IndexServer. Set to `true` to use SSL for the transport layer. Set to `False` to use the active normal mode, without SSL configuration. **Default value**: false **Possible or Recommended value**: true/false **Required**: No
PERSIST_LOGS	To persist the log files in `rtdata` volume or not. Log files including: `dsearch.log`, `dsearchadminweb.log`, `xdb.log`, `server.log`, `rest.log`, `cps.log` and `cps_daemon.log`. Set to `true` to keep log files in `/opt/xPlore/rtdata/DctmServer_node/logs`. Set to `false` to keep log files in `/opt/xPlore/logs`. **Default value**: true **Possible or Recommended value**: true/false **Required**: No

Parameter Name	Description
SCRIPT_DIR	The directory which is used for custom script. User must mount custom scripts to this directory. **Default value**: /opt/xPlore/rtdata/script **Required**: No
SCRIPT_INIT	The bash script file which is under the `script_dir` directory is executed the first time the container is started, after the configuration. **Default value**: init.sh **Required**: No
SCRIPT_BEFORE_START	The bash script file which is under the `script_dir` directory , is executed each time the container starts, before the JBoss start up. **Default value**: before-start.sh **Required**: No
EXT_CONF_PATH	External configuration path which is used to load the external configuration files. **Default value**: opt/xPlore/external-configurations **Possible or Recommended value**: opt/xPlore/external-configurations **Required**: No
CONFIG_NEW_RELIC	To check if configure new relic APM agent is present in the container. Set to `True`, the container automatically moves `newrelic.jar` (downloaded from the official website during the process of building image) to `$NEW_RELIC_ROOT`. The container doesn't contain the `configure yml,` file, you must load the `configure yml` to `$EXT_CONF_PATH` and when container is started, the `configuration yml` file is automatically copied to `$NEW_RELIC_ROOT`. **Default value**: false **Possible or Recommended value**: true/false **Required**: No
NEW_RELIC_ROOT	This is the path where the `newrelic.jar`, `newrelic.yml`, and the new relic log file are saved. It should not be set to `$EXT_CONF_PATH.` `/opt/xPlore/newrelic` **Required**: No

Parameter Name	Description
NEW_RELIC_APP_NAME_SUFFIX	The suffix of new relic application name, for one xPlore instance (including IndexServer, IndexAgent, and cps). It is recommended to set the same suffix value, but for different xPlore instances, it must be set to a special and unique value to avoid conflict. The actual application name will add `$NODE_NAME` (uppercase) as the prefix. **Required**: No

2.4.2 Environment parameters of IndexAgent

Parameter Name	Description
ESS_HOST	Index Server hostname. It should be the Index Server service name in K8s. **Default value**: indexserver **Required**: Yes
ESS_PORT	The port of the Index Server for communication. It can be set as `9300` or `9302`. To use with SSL configuration, set this field to `9302`. **Default value**: 9300 **Possible or Recommended value**: 9300 (normal), 9302 (ssl) **Required**: No
ESS_PROTOCOL	The communication protocol. It can be set as `http` or `https`. If using with SSL configuration, set this field to `https`. **Default value**: http **Possible or Recommended value**: http (normal). https (ssl) **Required**: No
MEM_ARGS	Part of `JAVA_OPTS`, bash script that starts the indexagent. `-XX:+UseCodeCacheFlushing -XX:ReservedCodeCacheSize=196m -Xms256m -Xmx1024m -Xss256k -XX:+DisableExplicitGC` **Possible or Recommended value**: copy the default value to modify **Required**: No

Parameter Name	Description
CONFIG_SSL	To set the SSL of the IndexAgent. Set to `true` to use SSL for the transport layer. Set to `false` to use the normal mode, without SSL configuration. **Default value**: false **Possible or Recommended value**: true/false **Required**: No
PERSIST_LOG	To persist the log files in `rtdata` volume or not. Log files including: `dsearchclient.log`, `indexagent.log` and `server.log`. Set as `true` to keep log files in `/opt/xPlore/rtdata/DctmServer_node/logs`; `false` to keep log files in `/opt/xPlore/logs`. **Default value**: true **Possible or Recommended value**: true/false **Required**: No
DOCBASE_NAM	Docbase name. **Required**: Yes
DOCBASE_USER	Docbase user. **Required**: Yes
DOCBASE_PASSWORD	Docbase password. **Required**: Yes
BROKER_HOST	Broker host.
BROKER_PORT	Broker port.
REGISTRY_NAME	Registry name.
REGISTRY_USER	Registry user.
REGISTRY_PASSWORD	Registry password.
IS_STANDBY	Is a standby mode index agent. **Default value**: false **Possible or Recommended value**: true/false **Required**: No

Parameter Name	Description
STANDBY_FULLTEXT_USER	Fulltext user in standby mode. **Default value**: dm_fulltext_index_user_01 **Required**: No
STANDBY_FULLTEXT_INDEX_NAME	Fulltext index name in standby mode. **Possible or Recommended value**: ${docbase_name}_ftindex_00 **Required**: No
STANDBY_FULLTEXT_ENGINE_NAME	Fulltext engine name in standby mode. **Possible or Recommended value**: DSearch Fulltext Engine Configuration 00 **Required**: No
SCRIPT_DIR	The directory which is used for custom script. User should mount custom scripts to this directory. **Default value**: /opt/xPlore/rtdata/script **Required**: No
SCRIPT_INIT	The bash script file that is under the directory `script_dir`. It is executed the first time the container starts after the configuration. **Default value**: init.sh **Required**: No
SCRIPT_BEFORE_START	The bash script file that is under the directory `script_dir`. It is executed each time the container starts after the JBoss start up. **Default value**: before-start.sh **Required**: No
SCRIPT_AFTER_START	The bash script file that is under the directory `script_dir`. It is executed each time the container starts before the JBoss start up. **Default value**: after-start.sh **Required**: No

Parameter Name	Description
FORCE_CREATE_OBJECT	Force create the Documentum Server object and the Index Server domain. The default logic is, when set to `false`, it creates Documentum Server object and index server domain when no data is found under volume (`/opt/xPlore/rtdata`). It will not create Documentum Server object and index server domain if the volume is not empty. Set it to `true` to override and force it to attempt to create objects. **Default value**: false **Possible or Recommended value**: true/false **Required**: No
EXT_CONF_PATH	External configuration path which is used to load the external configuration files. For external `dfc.properties`, only following properties are supported: `dfc.docbroker.host[N] (N=0, 1, 2....)` `dfc.docbroker.port[N]` `dfc.globalregistry.repository` `dfc.globalregistry.username`. **Default value**: opt/xPlore/external-configurations **Possible or Recommended value**: opt/xPlore/external-configurations **Required**: No
CONFIG_NEW_RELIC	To check if the configure new relic APM agent is present in the container. If the valude is set to true, the container will auto move `newrelic.jar` (which is downloaded from the official website during the process of building image) to `$NEW_RELIC_ROOT`. The container doesn't supply the `configure yml`, user should mount the `configure yml` to `$EXT_CONF_PATH` and when container is started, the `configuration yml` is copied automatically to `$NEW_RELIC_ROOT`. **Default value**: false **Possible or Recommended value**: true/false **Required**: No
NEW_RELIC_ROOT	The path in container to save `newrelic.jar` and `newrelic.yml`, and also the new relic log will be saved in this path. Ensure that it must not be `$EXT_CONF_PATH`. **Default value**: /opt/xPlore/newrelic **Required**: No

Parameter Name	Description
NEW_RELIC_APP_NAME_SUFFIX	The suffix of new relic application name, for one xPlore instance (including IndexServer, IndexAgent, and cps). It is recommended to set the same suffix value. For different xPlore instances, it must be set to a special and unique value to avoid conflict. The actual application name will add `$NODE_NAME` (uppercase) as it's prefix. **Required**: No

2.4.3 Environment parameters of CPS

Parameter Name	Description
ESS_HOST	Index Server hostname. **Possible or Recommended value**: indexserver **Required**: Yes
ESS_PORT	The port of the Index Server for communication. It can be set to `9300` or `9302`. If SSL is configured, set this field to `9302`. **Default value**: 9300 **Possible or Recommended value**: 9300 (normal) 9302 (ssl) **Required**: No
ESS_PASSWORD	The password of `dsearch`. **Default value**: password **Required**: No
ESS_PROTOCOL	The communication protocol; it can be set to `http` or `https`. If SSL is configured, set this field to `https`. **Default value**: http **Possible or Recommended value**: http (normal) https (ssl) **Required**: No

Parameter Name	Description
MEM_ARGS	Part of `JAVA_OPTS`, bash script for starting CPS. **Default value**: `-XX:+UseCodeCacheFlushing -XX:ReservedCodeCacheSize=196m -Xms4096m -Xmx4096m -Xss1024k -XX:+UseCompressedOops -XX:+DisableExplicitGC -XX:-ReduceInitialCardMarks -Djava.awt.headless=true` **Possible or Recommended value**: copy default value to modify. **Required**: No
CONFIG_SSL	To set the SSL of the CPS. Set to `true` to use SSL for the transport layer. Set to `false` to the active normal mode, without SSL the configuration. It can be easily configured without setting any parameters with prefix as `ssl_`. All the parameters with the prefix `ssl_` use the default value. If you are not familiar with the configuration of SSL, you can skip defining the parameters. **Default value**: false **Possible or Recommended value**: true/false **Required**: No
PERSIST_LOGS	To persist the log files in `rtdata` volume or not. Log files including: `cps_manager.log`, `cps_daemon.log`, and `server.log`. Set as `true` to keep the log files in `/opt/xPlore/rtdata/DctmServer_node/logs`. Set as `false` to keep the log files in `/opt/xPlore/logs`. **Default value**: true **Possible or Recommended value**: true/false **Required**: No
SCRIPT_DIR	The directory that is used for custom script. User must mount custom scripts to this directory. **Default value**: `/opt/xPlore/rtdata/script` **Required**: No

Parameter Name	Description
SCRIPT_INIT	The bash script file that is under the `script_dir`. It is executed the first time the container starts after the configuration. **Default value**: `init.sh` **Required**: No
SCRIPT_BEFORE_START	The bash script file that is under the `script_dir`. It is executed each time the container starts before the JBoss start up. **Default value**: `before-start.sh` **Required**: No
SCRIPT_AFTER_START	The bash script file that is under the directory `script_dir`. It is executed each time the container starts after the JBoss start up. **Default value**: `after-start.sh` **Required**: No
EXT_CONF_PATH	External configuration path which is used to mount external configuration files. `opt/xPlore/external-configurations` and `opt/xPlore/external-configurations`. **Required**: No
CONFIG_NEW_RELIC	To check if configure new relic APM agent is present in the container. If the valude is set to true, the container will automatically move the `newrelic.jar` (which is downloaded from the official website during the process of building image) to `$NEW_RELIC_ROOT`. The container does not supply the `configure yml`, the user must mount the `configure yml` to `$EXT_CONF_PATH` . When the container is started, the `configuration yml` file is automatically copied to `$NEW_RELIC_ROOT`. false **Possible or Recommended value**: true/false **Required**: No
NEW_RELIC_ROOT	The path in the container to save `newrelic.jar` , `newrelic.yml`, and the new relic log file. It must not be set to `$EXT_CONF_PATH`. **Default**: `/opt/xPlore/newrelic` **Required**: No
NEW_RELIC_APP_NAME_SUFFIX	The suffix of new relic application name, for one xPlore instance (including IndexServer, IndexAgent, cps). It is recommended to set the same suffix value, but for different xPlore instances, it must be set to a special and unique value to avoid conflict. The actual application name will add `$NODE_NAME` (uppercase) as prefix. No

2.5 Limitations

The limitations of xPlore docker images:

- xPlore docker images longer supports multi-node deployment.
- xPlore docker images no longer uses OIT and uses Tika instead as Text Extraction plugin.
- The CPS component is mandatory. The local CPS in indexserver is only used for searching, while remote CPSs are used for indexing.
- Uploading a local file is not supported in Admin UI.

2.6 Troubleshooting

There are no troubleshooting information for this release.

Chapter 3

Deploying xPlore on private Cloud

3.1 Overview

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

You can use the containerized deployment using the docker images and Helm Charts that are packaged with the release.

3.2 Prerequisites

Ensure that you complete the following activities before you deploy xPlore on private cloud:

1. Download and configure the Docker from the Docker website.
2. Download and configure the Kubernetes application from the Kubernetes website.
3. Download and configure the supported version of Helm package from the Helm website. The product Release Notes document contains detailed information about the supported versions. Helm Documentation contains detailed information.
4. Download the Docker image from OpenText Container Registry. Perform the following tasks:

    a. Log in to OpenText Container Registry using the following command format:

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

    When prompted, provide your OpenText My Support login credentials.

    b. Download the Docker image using the following command format:

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

    The following Docker images are available for download:

Image name	Image tag
dctm_xplore_indexserver	22.1.0
dctm_xplore_indexagent	22.1.0
dctm_xplore_cps	22.1.0

5. Download the Helm charts from OpenText My Support.

6. Download the Docker Compose file from OpenText My Support.

3.3 Deploying xPlore on private cloud

1. Load the xPlore images into the local Docker registry using the following command and update the exact xPlore images' name:

```
#docker load -i <TAR file name of the xPlore image>
```

2. Extract the Helm Charts TAR file to a temporary location.

3. Load the Graylog Docker image from the repository hub, upload this to your local repository and configure as appropriate.

4. Open the `xplore/values.yaml` file and provide the appropriate values for the variables depending on your environment to pass them to your templates as described in the following table:

Category	Name	Description
ingressDomain	ingressDomain	Specifies the `ingress domain` for the Ingress host configuration. The host format is: `ingressName.ingressDomain`.
ingressName	ingressName	Specifies the `ingress name` for the Ingress host configuration. The host format is `ingressName.ingressDomain`.
service	indexserver	• *<name>*: Specifies the IndexServer service name. • *<port>*: Specifies the IndexServer service port. • *<targetPort>*: Specifies the IndexServer service target port.
	indexagent	• *<name>*: Specifies the IndexAgent service name. • *<port>*: Specifies the IndexAgent service port. • *<targetPort>*: Specifies the IndexAgent service target port.
	cps	• *<name>*: Specifies the CPS service name. • *<port>*: Specifies the CPS service port. • *<targetPort>*: Specifies the CPS service target port.
image	repository	The image repository to pull the IndexServer, IndexAgent, and CPS images.

Category	Name	Description
	pullPolicy	The image pull policy of a pod container. The default is <*IfNotPresent*>, refer to the *Kubernetes documentation* for more details.
	indexserver	• <*name*>: The image name of IndexServer, it will use <*image.repository*> property as the prefix. • <*tag*>: The image tag of IndexServer.
	indexagent	• <*name*>: The image name of IndexAgent, it will use <*image.repository*> property as the prefix. • <*tag*>: The image tag of IndexAgent.
	cps	• <*name*>: The image name of CPS, it uses <*image.repository*> property as the prefix. • <*tag*>: The image tag of CPS.
graylog	enable	Set as true to create graylog sidecar container, set as false will skip
	image	The full image path of graylog. Note this value does not add <*image.repository*> property as the prefix.
	server	The graylog server IP address.
	port	The graylog server IP port.
indexserver	persistLogs	Set as `false` to use graylog, set as `true` to persist logs in the volume.

Category	Name	Description
	extraEnv	The extra environment parameters that are passed to indexserver container: • *<script_dir>*: The directory which is used for custom script. This property is disabled in Helm chart and default value is *</opt/xPlore/rtdata/script>*(you must mount the script folder to this path). To enable it, uncomment it and related lines in `indexserver-statefulset.yaml`. You must also mount the script folder to this path. • *<script_init>*The bash script file which is within scriptDir directory will be executed the first time you start the container, after the configuration. This property is disabled in Helm chart and default value is *<init.sh>*. To enable it, uncomment it and related lines in `indexserver-statefulset.yaml`. • *<script_before_start>*: The bash script file which is within the scriptDir directory will be executed each time you start the container, before JBoss start up. This property is disabled in Helm chart and default value is *<before-start.sh.>* To enable it, uncomment it and related lines in `indexserver-statefulset.yaml`. • *<script_after_script>*: The bash script file which is within the scriptDir directory will be executed each time you start the container, after JBoss start up. This property is disabled in Helm chart and default value is *<after-start.sh.>* To enable it, uncomment it and related lines in `indexserver-statefulset.yaml`. • MEM_ARGS: -XX:+UseCodeCacheFlushing -XX:ReservedCodeCacheSize=196m -Xms1024m -Xmx2048m -Xss512k -XX:+DisableExplicitGC
	livenessProbe	• *<initialDelay>*: Specifies the livenessProbe initialDelaySeconds value of IndexServer pod container. • *<timeout>*: Specifies the livenessProbe timeoutSeconds value of IndexServer pod container. • *<period>*: Specifies the livenessProbe periodSeconds value of IndexServer pod container.

Category	Name	Description
	readinessProbe	• <*initialDelay*>: Specifies the readinessProbe initialDelaySeconds value of IndexServer pod container. • <*timeout*>: Specifies the readinessProbe timeoutSeconds value of IndexServer pod container. • <*period*>: Specifies the readinessProbe periodSeconds value of IndexServer pod container.
indexagent	essHost	The IndexServer host, default value is IndexServer service name.
	brokerHost	Documentum Server broker host.
	brokerPort	Documentum Server broker port.
	docbaseName	Documentum Server docbase name.
	docbaseUser	Documentum Server docbase user.
	docbasePassword	Documentum Server docbase password.
	registryName	Documentum Server registry name.
	registryUser	Documentum Server registry user.
	registryPassword	Documentum Server registry password.
	persistLogs	Set it as 'false' to use graylog, otherwise 'true' to persist logs in the volume.

Category	Name	Description
	extraEnv	The extra environment parameters that are passed to indexserver container: • *<force_create_object>*: The flag is used to decide whether the IndexAgent will force create related objects in Documentum database or not. The default value is `true`. • *<script_dir>*: The directory which is used for custom script. This property is disabled in Helm chart and default value is *</opt/xPlore/rtdata/script>*(you should mount the script folder to this path). To enable it, uncomment it and related lines in `indexagent-statefulset.yaml`. • *<script_init>*: The bash script file in `scriptDir` directory is executed the first time the container is started after the configuration. This property is disabled in helm chart and default value is `init.sh`. To enable it, uncomment it and related lines in `indexagent-statefulset.yaml`. • *<script_before_start>*: The bash script file in the `scriptDir` directory is executed each time the container starts, before JBoss start up. This property is disabled in Helm chart and default value is *<before-start.sh>*. To enable it, uncomment it and related lines in `indexagent-statefulset.yaml`. • *<script_after_start>*: The bash script file which is within the scriptDir directory will be executed each time you start the container, after JBoss start up. This property is disabled in Helm chart and default value is *<after-start.sh>*. To enable it, uncomment it and related lines in `indexagent-statefulset.yaml`. • Index Agent Container memory arguments are available as parameters for the users can pass the values: *<MEM_ARGS: -XX:+UseCodeCacheFlushing -XX:ReservedCodeCacheSize=196m -Xms1024m -Xmx2048m -Xss512k -XX:+DisableExplicitGC>*

Category	Name	Description
	livenessProbe	• <*initialDelay*>: Specifies the livenessProbe initialDelaySeconds value of the IndexAgent pod container. • <*timeout*>: Specifies the livenessProbe timeoutSeconds value of the IndexAgent pod container. • <*period*>: Specifies the livenessProbe periodSeconds value of the IndexAgent pod container.
	readinessProbe	• <*initialDelay*>: Specifies the readinessProbe initialDelaySeconds value of the IndexAgent pod container. • <*timeout*>: Specifies the readinessProbe timeoutSeconds value of the IndexAgent pod container. • <*period*>: Specifies the readinessProbe periodSeconds value of the IndexAgent pod container.
cps	replicaCount	The replicas number of CPS StatefulSet. The default value is 2.
	essHost	The IndexServer host, default value is IndexServer service name.
	persistLogs	Set as `false` to use graylog, Set as `true` to persist logs in the volume.

Category	Name	Description
	extraEnv	The extra environment parameters that are passed to indexserver container: • *<register_cps_delay>*: A number value of time (unit : second), which is used for delay `add_cps` execution. This execution will register CPS to IndexServer. The recommended value should be greater than the value of *<cps.readinessProbe.initialDelay>*. • `script_dir`: The directory which is used for custom script. This property is disabled in Helm chart and default value is *</opt/xPlore/rtdata/script>*(you must mount the script folder to this path). To enable , uncomment it and the related lines in `cps-statefulset.yaml`. • `script_init`: The bash script file in the `scriptDir` directory is executed the first time the container starts after the configuration. This property is disabled in Helm chart and default value is *<init.sh.>* To enable it, uncomment it and related lines in `cps-statefulset.yaml`. • `script_before_start`: The bash script file in the `scriptDir` directory is executed each time you start the container, before JBoss start up. This property is disabled in Helm chart and default value is *<before-start.sh>*. To enable it, uncomment it and related lines in `cps-statefulset.yaml`. • `script_after_start`: The bash script file in the `scriptDir` directory is executed each time you start the container, after JBoss start up. This property is disabled in Helm chart and default value is*<after-start.sh>*. To enable it, uncomment it and related lines in `cps-statefulset.yaml`.
	livenessProbe	• *<initialDelay>*: Specifies the livenessProbe initialDelaySeconds value of CPS pod container. • *<timeout>*: Specifies the livenessProbe timeoutSeconds value of CPS pod container. • *<period>*: Specifies the livenessProbe periodSeconds value of CPS pod container.

Category	Name	Description
	readinessProbe	• <*initialDelay*>: Specifies the readinessProbe initialDelaySeconds value of CPS pod container. • <*timeout*>: Specifies the readinessProbe timeoutSeconds value of CPS pod container. • <*period*>: Specifies the readinessProbe periodSeconds value of CPS pod container.
volumeMounts	subPath	Specifies the sub path of volume mounts.
persistentVolume	isExist	Specifies if an existing PVC is used. The default value is `false` and will create a new one in helmchart.
	claimName	Specifies the claimName value of PVC.
	accessModes	Specifies the accessModes value of PVC.
	size	Specifies the resources.requests.storage value of PVC.
	storageClass	Specifies the storageClassName value of PVC.
customVolume	customMountPath	Specifies the mount path in container when user mounts a custom PVC
	customClaimName	Specifies the claimName value of custom PVC
resources	indexserver	• <*requests.memory*>: Specifies the initial memory resource of IndexServer pod container. • <*requests.cpu*>: Specifies the initial cpu resource of IndexServer pod container. • <*limits.memory*>: Specifies the maximal memory resource of IndexServer pod container. It is recommended to have a value greater than 6144Mi. • <*limits.cpu*>: Specifies the maximal cpu resource of IndexServer pod container. It is recommended to have a value greater than 2000m.

Category	**Name**	**Description**
	indexagent	• *<requests.memory>*: Specifies the initial memory resource of IndexAgent pod container. • *<requests.cpu>*: Specifies the initial cpu resource of IndexAgent pod container. • *<limits.memory>*: Specifies the maximal memory resource of IndexAgent pod container. It is recommended to have a value greater than 4096Mi. • *<limits.cpu>*: Specifies the maximal cpu resource of IndexAgent pod container. It is recommended to have a value greater than 200m.
	cps	• *<requests.memory>*: Specifies the initial memory resource of CPS pod container. • *<requests.cpu>*: Specifies the initial cpu resource of CPS pod container. • *<limits.memory>*: Specifies the maximal memory resource of CPS pod container. It is recommended to have a value greater than 6144Mi. • *<limits.cpu>*: Specifies the maximal cpu resource of CPS pod container. It is recommended to have a value greater than 2500m.
useSecrets	useSecrets	Using secrets to pass Content Server related passwords. The default value is `true`.
secrets	cs	• *<name>*: The secret resource name to be used. • *<key>*: The key name which is used in secret resource object, xPlore uses: *<docbasePassword>*installOwnerPassword *<registryPassword>* globalRegistryPassword
useConfigMap	useConfigMap	Use configmap to provide `dfc.properties`, the default value is `true`.
configMap	cs	• *<name>*: The configmap name for dfc. • *<key>*: The key name which is used in configmap. The default value is :*<dfcPropertiesKey>*: dfc.properties

5. Deploy the xPlore Helm in your private cloud.
```
helm install <location where Helm Chart TAR files are extracted>/
xplore --name <release name>
```
For example:
```
helm install /opt/temp/Helm-charts/xplore --name xplore
```

6. Verify the status of xPlore Helm deployment.`helm status <release name>`.

3.4 Upgrading xplore on private cloud

1. Before upgrading the xPlore in private cloud, ensure that you have default replica value for each component.

2. Update the new image details in `xplore/values.yaml` for the xPlore Helm Chart and upgrade using the following command format:

   ```
   helm upgrade <release name>./xplore
   ```

   Notes

   - Upgrade of image is supported. You must update image-related values in xplore/values.yaml only. All other values must not be changed.
   - Upgrade process is in descending order. xPlore indexserver and indexagent only support replica being 1. So indexserver and indexagent will be down for several minutes during upgrading.
   - If you encounter any problems during the upgrade process with the new image, then the upgrade process stops automatically. Also, you can roll back to the previous image.
   - Upgrade process takes approximately 10 minutes totally.

3. Verify the status of the successful upgrade using the following steps:

   a. Check the status of the pod. If the upgrade is successful, the status of the pod is active.

   b. Check xPlore version. Log in to xPlore Administrator web console, click **System Overview**. If the upgrade is successful, then the new xPlore version is displayed.

3.5 Rolling back the upgrade process

Rolling back the upgrade process (rolling back to the previous image) is recommended when the upgrade process fails or when you encounter errors using the new image.

Perform the following steps:

1. Fetch the details of history using the following command format:

   ```
   helm history <release name>
   ```

2. Roll back to the previous image using the following command format:

   ```
   helm rollback <release name> <version>
   ```

3.6 Limitations

- xPlore removes OIT and uses Tika instead as Text Extraction plugin.
- CPS component is mandatory.
- IndexServer and IndexAgent does not support replica.
- The limits of the resources should not be smaller than the default values in xPlore helmchart.
- Uploading the Testing Document for the local file in the xPlore administrator user interface is not supported in the cloud environment.
- Installation owner is predefined and cannot be changed. The value is `dmadmin`.
- Installation path is predefined and cannot be changed. The value is `/root/xPlore`.

3.7 Troubleshooting

There are no troubleshooting information for this release.

Chapter 4

Deploying xPlore on Microsoft Azure cloud platform

4.1 Prerequisites

Ensure that you complete the following activities before you deploy xPlore on Azure environment:

1. Download and configure the Docker application from the Docker website.

   Docker documentation contains detailed information.

2. Download and configure the supported version of Helm package from the Helm website. The product Release Notes document contains detailed information about the supported versions.

   Helm documentation contains detailed information.

3. Download and configure the Kubernetes application from the Kubernetes website. The product Release Notes document contains detailed information about the supported versions.

   Kubernetes documentation contains detailed information.

4. Set up Azure.

   a. Create a resource group. A resource group in Azure is a folder to keep your collection. It does not serve any other purpose.

      Navigate to **Home > Resource groups > Resource group** and provide the following information:

      - **Resource group name**
      - **Subscription**
      - **Resource group location**

      Click **Create**.

   b. Create a container registry. Container registry is used to store the Docker images in Azure. A standard container registry can store up to 100 GB of images.

      Navigate to **Home > Container registries > Create container registry** and provide the following information:

      - **Registry name**
      - **Subscription**
      - **Resource group**
      - **Location**

- **Admin user**
- **SKU**

Click **Create**.

c. Create an Azure Kubernetes Service (AKS). AKS is a managed container orchestration service, based on the open source Kubernetes system, which is available on the Azure public cloud.

Navigate to **Home > Kubernetes services > Create Kubernetes cluster** and provide the information in the following tabs:

- **Basics**: Provide valid values for all the mandatory fields such as **PROJECT DETAILS**, **CLUSTER DETAILS**, and so on. Select the **F8** with the family as Memory Optimized for the virtual machine size.

  Click **Next: Authentication >**.

- **Authentication**: Provide valid values for all the mandatory fields such as **CLUSTER INFRASTRUCTURE** and **KUBERNETES AUTHENTICATION AND AUTHORIZATION**. Enable Role-based access control (RBAC).

  Click **Next: Networking >**.

- **Networking**: Disable **HTTP application routing** and set the proper ingress controller. Also, select **Basic** for **Network configuration**.

  Click **Next: Monitoring >**.

- **Monitoring**: Enable the container monitoring. Also, select the log analytics workspace.

  Click **Next: Tags >**.

- **Tags**: (Optional) Name or value pairs that enable you to categorize resources.

  Click **Next: Review + create >**.

- **Review + create**: Review the summary of information and click **Create** to create an AKS.

5. Configure Azure on Linux VM using the following commands:

```
az aks install-cli
sudo yum update azure-cli
sudo sh -c 'echo -e "[azure-cli]\nname=Azure
CLI\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
azurecli\nenabled=1\ngpgcheck=1\
ngpgkey=https://packages.microsoft.com/keys/microsoft.asc">
/etc/yum.repos.d/azure-cli.repo'
sudo yum install azure-cli
az aks install-cli
az aks get-credentials --resource-group dctm --name dctmaks
az login
az login -u <id>@opentext.com
```

After the configuration, view the configuration of Azure using the following command format:

```
kubectl config view
```

Example output:

```
[root@skvcentos ~]# kubectl config view
apiVersion: v1
clusters:
- cluster:
certificate-authority-data: DATA+OMITTED
server: https://dctmaks-382bfe75.hcp.eastus.azmk8s.io:443
name: dctmaks
contexts:
- context:
cluster: dctmaks
user: clusterUser_dctm_dctmaks
name: dctmaks
current-context: dctmaks
kind: Config
preferences: {}
users:
- name: clusterUser_dctm_dctmaks
user:
client-certificate-data: REDACTED
client-key-data: REDACTED
token: 0506b0a1bd38280fc155a15ae2641eb8
```

6. Configure the Helm package on Linux VM. Perform the following tasks:

   a. Extract the Helm package you downloaded to a temporary location.

   b. Locate the Helm binary and move it to the `/usr/local/bin/helm` folder using the following command format:

   ```
   mv linux-amd64/helm /usr/local/bin/helm
   ```

7. Retrieve the details of the resource using the following command format:

```
az aks show --resource-group <resource group name>
--name <name of the cluster>
--querynodeResourceGroup -o tsv
```

Example output:

```
[root@skvcentos ~]# az aks show --resource-group dctm
--name dctmaks
--query nodeResourceGroup -o tsv
MC_dctm_dctmaks_eastus
[root@skvcentos ~]#
```

8. Create a storage account using the following command format:

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

Example output:

```
[root@skvcentos ~]# az storage account create
--resource-group MC_dctm_dctmaks_eastus
name dctmstorageacc --sku Standard_LRS
{
"accessTier": null,
"creationTime": "2018-11-26T06:11:58.380457+00:00",
"customDomain": null,
"enableHttpsTrafficOnly": false,
"encryption": {
"keySource": "Microsoft.Storage",
"keyVaultProperties": null,
```

```
"services": {
"blob": {
...
...
[root@skvcentos ~]#
```

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

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

   Different types of storage are:

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

   a. Create a storage class using the following command format:

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

   b. Apply the configuration to a resource using the name of the YAML file using the following command format:

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

   Example output:

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

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

10. Create the cluster role, cluster binding, a YAML file (for example, `azure-pvc-roles.yaml` with appropriate values for the parameters), and apply the configuration.

    AKS clusters use Kubernetes RBAC to limit actions that can be performed. Roles define the permissions to grant and bindings apply them to the desired users.

    a. Create the cluster role using the following command format:

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
name: system:azure-cloud-provider
rules:
- apiGroups: ['']
resources: ['secrets']
verbs: ['get','create']
```

b. Create the cluster binding using the following command format:

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
name: system:azure-cloud-provider
roleRef:
kind: ClusterRole
apiGroup: rbac.authorization.k8s.io
name: system:azure-cloud-provider
subjects:
- kind: ServiceAccount
namespace: kube-system
name: persistent-volume-binder
```

c. Apply the configuration to a resource using the name of the YAML file using the following command format:

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

Example output:

```
[root@skvcentos ~]# kubectl apply -f azure-pvc-roles.yaml
clusterrole.rbac.authorization.k8s.io/system:azure-cloud-provider
created
clusterrolebinding.rbac.authorization.k8s.io/system:azure-cloud-provider
created
[root@skvcentos ~]#
```

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

11. Create a persistent volume claim, a YAML file (for example, `azure-pvc-roles.yaml` with appropriate values for the parameters), and apply the configuration.

A persistent volume claim (PVC) uses the storage class object to dynamically provision an Azure file share.

Create a persistent volume claim using the following command format:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: nfs-volume-claim  (refers to volume claim)
spec:
accessModes:
- ReadWriteMany
storageClassName: azurefile
resources:
requests:
storage: 100Gi
```

Example output:

```
[root@skvcentos ~]# kubectl apply -f azure-file-pvc.yaml
persistentvolumeclaim/nfs-volume-claim created

[root@skvcentos ~]# kubectl get pvc
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
```

```
nfs-volume-claim Pending azurefile 5s

[root@skvcentos ~]# kubectl get pvc
NAME STATUS VOLUME CAPACITY
ACCESS MODES STORAGECLASS AGE
nfs-volume-claim Bound pvc-2b17c86b-f144-11e8-a3d7-828b27cd3822 100Gi
RWX azurefile 10s
[root@skvcentos ~]#
```

12. Create new role assignment for a user, group, or service principal using the following command format:

    ```
    az role assignment create
    --assignee $CLIENT_ID
    --role Reader
    --scope $ACR_ID
    ```

    Example output:

    ```
    az role assignment create
    --assignee 24a416a4-2312-47af-ab54-ccfaa333b688
    --role Reader
    --scope /subscriptions/c0b06162-314b-449a-9c97-
    c7f8645e8cc0/resourceGroups/dctm/providers/
    Microsoft.ContainerRegistry/registries/dctmcr
    ```

13. Load the image into Azure.

    a. Verify if Docker daemon or server is running on the local system or server using the following command format:

    ```
    sudo /usr/bin/dockerd --insecure-registry
    <registry_ip : registry_port>
    --insecure-registry <registry_ip : registry_port>
    -H unix:///var/run/docker.sock --init 2>&1 &
    ```

    Example output:

    ```
    sudo /usr/bin/dockerd --insecure-registry 10.194.42.173:5000
    --insecure-registry 10.8.176.180:5000
    --insecure-registry 10.8.146.181:80
    --insecure-registry 10.8.176.180:5000
    --insecureregistry 10.9.56.50:80
    --insecure-registry dctmcr.azurecr.io
    --insecure-registry 10.9.57.7 -H unix:///var/run/docker.sock
    --init 2>&1 &
    ```

    b. Download or pull the image into the local registry using the following command format:

    ```
    docker pull
    <artifactory.otxlab.net/bpdockerhub/dctm-xplore-indexserver:22.1.0>
    ```

    Example output:

    ```
    [root@csazure ~]# docker pull
    artifactory.otxlab.net/bpdockerhub/dctm-xplore-indexserver:22.1.0
    256b176beaff: Pull complete
    fff58857e6a6: Pull complete
    8180823e9cbd: Pull complete
    b0b2b563b195: Pull complete
    26aa433e4325: Pull complete
    22d5939fd717: Pull complete
    4cc820e77a67: Pull complete
    fbf522c2ac97: Pull complete
    0e3bfa118948: Pull complete
    ad0a41aa39d0: Pull complete
    c6256484ce4f: Downloading [===>
    ```

```
112.1MB/1.482GB
1bbeba7bc6bb: Download complete
```

c. Tag the image to the Azure registry specific image using the following command format:

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

Example output:

```
[root@csazure ~]# docker pull
artifactory.otxlab.net/bpdockerhub/dctm-xplore-indexserver:22.1.0
dctmcr.azurecr.io/bpdockerhub/dctm-xplore-indexserver:22.1.0
```

d. Log in the Azure container registries if not already logged in using the following command format:

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

Example output:

```
[root@csazure ~]# az acr login --name dctmcr
Login Succeeded
WARNING! Your password will be stored unencrypted in
/root/.docker/config.json.
Configure a credential helper to remove this warning.
[root@csazure ~]#
```

e. Push the image to the Azure Container Registry (ACR) using the following command format:

```
docker push <image>
```

Example output:

```
[root@csazure ~]# docker push
dctmcr.azurecr.io/bpdockerhub/dctm-xplore-indexserver:22.1.0
The push refers to repository
[dctmcr.azurecr.io/xplore/centos/stateless/indexserver]
55d9a89a5421: Pushed
2dea5cbbbcc2: Pushing [===>]
1.002GB/2.062GB
6bd9de9900c6: Pushed
d67c7dc8673c: Pushed
255f0292e611: Pushed
fa7f7298db17: Pushed
05e72c68ac14: Layer already exists
e448fb1da7d5: Layer already exists
5c1bb4d51f07: Layer already exists
730720397169: Layer already exists
0bf1949d4323: Layer already exists
1d31b5806ba4: Layer already exists
```

14. Download the Docker image from OpenText Container Registry. Perform the following tasks:

a. Log in to OpenText Container Registry using the following command format:

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

When prompted, provide your OpenText My Support login credentials.

b. Download the Docker image using the following command format:

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

The following Docker images are available for download:

Image name	Image tag
dctm_xplore_indexserver	22.1.0
dctm_xplore_indexagent	22.1.0
dctm_xplore_cps	22.1.0

15. Download the Helm charts from OpenText My Support.
16. Deploy Documentum Server as described in *OpenText Documentum Platform and Platform Extensions Cloud Deployment Guide*.

4.2 Deploying xPlore

1. Extract the Helm Charts TAR file to a temporary location.
2. Open the `xplore/values.yaml` file and provide the appropriate values for the variables depending on your environment to pass them to your templates. For your reference, see "Deploying xPlore on private cloud" on page 22 the for the descriptions of the parameters.
3. Deploy the xPlore Helm in your private cloud using the following command format:

   ```
   helm install <location where Helm Chart TAR files are extracted> -name<your defined
   release name>
   ```

   Example:

   ```
   helm install /opt/temp/Helm-charts/xplore --name xplore
   ```

4. Verify the status of the deployment of xPlore Helm using the following command format:

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

5. Verify the status of the deployment of xPlore pod using the following command format:

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

4.3 Configuring external IP address

1. Install the ingress controller that uses <*ConfigMap*> to store the nginx configuration using the following command format:

   ```
   helm install stable/nginx-ingress --namespace <name of the namespace>
   ```

   Ingress controller is an assembly of a configuration file (`nginx.conf`). The main requirement is the need to reload nginx after you make any changes to the configuration file. For example:

   ```
   apiVersion: extensions/v1beta1
   kind: Ingress
   metadata:
   name: xplore-ingress
   annotations:
   nginx.ingress.kubernetes.io/affinity: "cookie"
   ```

```
nginx.ingress.kubernetes.io/session-cookie-hash: "sha1"
spec:
rules:
- host: xpl-aks-ingress.eastus.cloudapp.azure.com
<dns name generated with external IP address>
http:
paths:
- backend:
serviceName: indexserver
servicePort: 9300
path: /dsearchadmin
- backend:
serviceName: indexserver
servicePort: 9300
path: /dsearch
- backend:
serviceName: indexserver
servicePort: 9200
path: /IndexAgent
```

2. Configure an FQDN for the public IP address of your ingress controller. Map the external IP address to the DNS name using the Bash script. Use the following command format:

```
#!/bin/bash

# Public IP address of your ingress controller
IP="<IP address>"

# Name to associate with public IP address
DNSNAME="<dns name>"

# Get the resource-id of the public ip
PUBLICIPID=$(az network public-ip list
--query "[?ipAddress!=null]|[?contains(ipAddress, '$IP')].[id]"
--output tsv)

# Update public ip address with DNS name
az network public-ip update --ids $PUBLICIPID --dns-name $DNSNAME
```

4.4 Limitations

- Host name must have the fully qualified domain name (FQDN) and must not be greater than 59.
- You must change the storage class as per the Azure Kubernetes service offering. The *default* storage class provisions a standard Azure disk while the *managed-premium* storage class provisions a premium Azure disk.

4.5 Troubleshooting

There are no troubleshooting information for this release.

Chapter 5

Deploying Documentum xPlore on Google Cloud Platform

5.1 Prerequisites

Ensure that you complete the following activities before you deploy Documentum xPlore on Google Cloud Platform (GCP) environment:

1. Download and configure the Docker application from the Docker website. Docker Documentation contains detailed information.
2. Download and configure the supported version of Helm package from the Helm website. The product Release Notes document contains detailed information about the supported versions. Helm Documentation contains detailed information.
3. Download and configure the Kubernetes application from the Kubernetes website.

   The product Release Notes document contains detailed information about the supported versions.
4. Download the latest version of Google Cloud SDK from GCP website and install on the machine that includes gcloud command line utility.
5. Create a Google Kubernetes Engine (GKE) and a namespace within the cluster.
   1. From the GCP website, select the GCP project linked with your corporate billing account.
   2. Create a cluster. Navigate to **Kubernetes Engine**, then **Clusters**, click **CREATE CLUSTER**, and perform the following:
      a. Select a standard cluster template.

      b. Provide a name to your cluster.

      For example, demo-cluster.

      c. Select **Zonal** for the location type. You can use Regional for production grade-high available clusters.

      d. Select a Zone closer to your location. The Google Cloud Platform website contains the list of zones.

      e. Review the number of nodes for your cluster and the machine type for each of your nodes.

      For example, three nodes with 2vCPUs/7.5 GB memory for each node.

      f. Click **Create**.

3. Click **Connect** next to the cluster you created and then click **Run in Cloud Shell**. A Google Cloud shell (a VM created by Google with pre-installed Kubectl and gcloud SDK) is created.
4. Click Enter key at the command prompt that is displayed. Cluster credentials are fetched and creates an entry (the kubectl configuration file).
5. Create a Kubernetes cluster namespace on the Cloud shell using the following command format:

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

6. Verify if the namespace is created using the following command format:

```
kubectl get namespace
```

6. After downloading and installing Helm, perform the following:

1. Verify if the status of the deployed pod in the namespace using the following command format:

```
kubectl get pods -n <name of namespace>
```

Example Output:

```
NAME READY STATUS RESTARTS AGE
xolore1cps-deploy-567b658988-cj8ff 1/1 Running 0 27h
```

2. Verify the status of the deployed pods in the namespace using the following command format:

```
helm version
```

Example Output:

```
version.BuildInfo{Version:"v3.6.1",
GitCommit:"61d8e8c4a6f95540c15c6a65f36a6dd0a45e7a2f",
GitTreeState:"clean", GoVersion:"go1.16.5"}
```

7. Create a Google Cloud Filestore instance.

In Filestore of Google Cloud Platform dashboard, click CREATE INSTANCE, and perform the following:

- Provide a name to the Google Cloud Filestore instance. For example,

```
demogcfss
```

- Select a standard cluster template.
- Select the default authorized network.
- Select your region and zone for the Location type for better performance.
- Select the NFS mount point for the Fileshare name. For example, demogcfs.
- Provide a minimum of 1 TB for Fileshare Capacity. Google Cloud Filestore need not be created every time. The Google Cloud Filestore instance can be shared by multiple clusters. After the Google Cloud Filestore instance is created, click the instance and note the IP address and path of the instance.

Note: The instance tier of the Google Cloud Filestore instance cannot be modified once it is created. However, the Fileshare Capacity can be modified.

8. Deploy an external storage provisioner (nfs-client-provisioner) for dynamic provisioning of ReadWriteMany (RWX) Persistent Volumes (PVs) on Google Cloud Filestore instance.

   1. Download the external nfs-client-provisioner Helm chart from the Github website to your Cloud Shell machine.

   2. Install the nfs-client-provisioner Helm chart by passing the nfs.server value as the IP address of the Google Cloud Filestore instance, the nfs.path value as the path given in the Google Cloud Filestore instance and storageClass.name value.

      For example, gcp-rwx. ReadWriteMany Persistent Volume Claims must be created with this storageClassName for ReadWriteMany Persistent Volumes.

      For example:.

```
helm install --name demo-nfs-client-provisioner ./nfs-client-provisioner/ --
namespace demo --set nfs.server="10.226.6.226" --set nfs.path=/demogcfs --set
storageClass.name=gcp-rwx
```

      Example Output:

```
NAME:   demo-nfs-client-provisioner
LAST DEPLOYED: Sun Jun  9 11:52:05 2019
NAMESPACE: demo
STATUS: DEPLOYED

RESOURCES:
==> v1/Deployment
NAME                         AGE
demo-nfs-client-provisioner  1s

==> v1/Pod(related)

NAME                                          READY  STATUS
RESTARTS  AGE
demo-nfs-client-provisioner-76986844cf-g6dvw  0/1    ContainerCreating
0         1s

==> v1/StorageClass

NAME     AGE
gcp-rwx  1s

==> v1/ServiceAccount
demo-nfs-client-provisioner  1s

==> v1/ClusterRole
demo-nfs-client-provisioner-runner  1s

==> v1/ClusterRoleBinding
run-demo-nfs-client-provisioner  1s

==> v1/Role
leader-locking-demo-nfs-client-provisioner  1s

==> v1/RoleBinding
leader-locking-demo-nfs-client-provisioner  1s
```

9. Create a sample PVC , for example, `testPVC.yaml` to test the dynamic provisioning of ReadWriteMany (RWX) PVs using the following command format:

   ```
    kubectl create -f testPVC.yaml
   ```

   This is the sample content of the `testPVC.yaml`:

   ```
   kind: PersistentVolumeClaim
   apiversion: v1
   metadata:
   name: test-claim
   spec:
   accessModes:
   - ReadWriteMany
   storageClassName:
   gcp-rwx
   resources:
   requests:
   storage: 1Mi
   ```

10. Verify if the corresponding Persistent Volume is created using the following command format:

    ```
    kubectl get pv
    ```

    Example output:

    ```
    NAME                                       CAPACITY   ACCESS MODES   RECLAIM
    POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
    pvc-7ed2d89f-8a7f-11e9-824b-42010aa0006f   1Mi        RWX
    Delete           Bound    demo/test-claim   gcp-rwx                 66s
    ```

    - Ensure the status of PV is Bound.
    - Delete the PVC and verify if the PV is deleted.

11. Download the Docker image from OpenText Container Registry. Perform the following tasks:

    a. Log in to OpenText Container Registry using the following command format:

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

    When prompted, provide your OpenText My Support login credentials.

    b. Download the Docker image using the following command format:

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

    The following Docker images are available for download:

Image name	Image tag
dctm_xplore_indexserver	22.1.0
dctm_xplore_indexagent	22.1.0
dctm_xplore_cps	22.1.0

12. Download the Helm charts from OpenText My Support.

13. Deploy Documentum Server as described in *OpenText Documentum Platform and Platform Extensions Cloud Deployment Guide*.

5.2 Deploying Documentum xPlore on Google Cloud Platform

1. Extract the Helm Charts TAR file to a temporary location.

2. Upload the Documentum xPlore Docker Images to Google Container Registry (GCR).

   a. Configure Docker to use gcloud as a credential helper using the following command format:

   ```
   gcloud auth configure-docker
   ```

   b. Tag your Docker images using the following command format:

   ```
   [HOSTNAME]/[PROJECT-ID]/[IMAGE]
   ```

   For example:

   ```
   docker push gcr.io/documentum-search-product/xplore/centos/stateless/
   indexagent:16.7.0000.0343
   ```

   ```
   gcr.io/documentum-search-product/xplore/centos/stateless/indexagent:
   16.7.0000.0343
   ```

   c. Load the tagged image to GCR

   For example:

   ```
   docker push gcr.io/documentum-search-product
   /xplore/centos/stateless/indexagent:16.7.0000.0343
   ```

3. Install the Documentum xPlore Helm Charts.

   a. Update the image and tag the fields in the `values.yaml` files of the Helm charts to point to the GCR images.

   For example:

   ```
   ### Docker Image ###
   image:
    repository: gcr.io
    pullPolicy: IfNotPresent
    indexserver:
    name: documentum-search-product/xplore/centos/stateless/indexserver
    tag: 16.7.0000.0343
    indexagent:
    name: documentum-search-product/xplore/centos/stateless/indexagent
    tag: 16.7.0000.0343
    cps:
    name: documentum-search-product/xplore/centos/stateless/cps
    tag: 16.7.0000.0343
   ```

   b. Update the storageClass fields in the `values.yaml` with storageClass : standard for ReadWriteOnce PVC and storageClass : gcp-rwx for ReadWriteMany PVC.

   When creating the nfs-client-provisioner, the storage class name created is gcp-rwx.

Update other fields in the `values.yaml` and install the Helm charts.

```
### Persistent Volume ###
volumeMounts:
 subPath: "."
persistentVolume:
 isExist: false
 claimName: xpl-data-pvc
 # If isExist is true, then accessModes,size,storageClass will not be
effective.
 accessModes: ReadWriteMany
 size: 1Gi
 storageClass: gcp-rwx
```

c. xPlore Helm Chart installation:

   i. Open the `values.yaml` file of xPlore Helm Chart and provide the appropriate values for the variables depending on your environment to pass them to your templates.

   ii. Deploy the Documentum xPlore Helm Chart in your GCP using the following command format:

   ```
   helm install <location where Helm Char Tar files are extracted> /xplore
    --name <release name>
   ```

   For example:

   ```
   helm install ../HelmCharts_xPlore/ --namespace demo --name xplore-gcp
   ```

   iii. Verify the status of the deployment of Documentum Server Helm using the following command format:

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

   iv. Verify the status of the deployment of Documentum xPlore pod using the following command format:

   ```
    kubectl describe pods
   ```

4. Install NGINX (Ingress Controller) Helm Chart:

   a. NGINX Ingress consists of two components:

   - Ingress Resource: Collection of rules for the inbound traffic to reach Services. These are Layer 7 (L7) rules that allow hostnames (and optionally paths) to be directed to specific Services in Kubernetes.
   - Ingress Controller: Acts upon the rules set by the Ingress Resource, typically via an HTTP or L7 load balancer.

   b. Deploy the NGINX Ingress Controller Helm using the following command format:

   ```
   helm install --name<release name> stable/nginx-ingress --set rbac.create=true
   --namespace<name of namespace>
   ```

   For example:

   ```
   helm install --name demo-nginx-ingress stable/nginx-ingress --set
   rbac.create=true --namespace demo
   ```

   Example Output:

```
NAME:   demo-nginx-ingress
LAST DEPLOYED: Sun Jun  9 15:48:55 2019
NAMESPACE: demo
STATUS: DEPLOYED

RESOURCES:
==> v1beta1/ClusterRoleBinding
NAME                AGE
demo-nginx-ingress  0s

==> v1beta1/Role
demo-nginx-ingress  0s

==> v1/Service
demo-nginx-ingress-controller       0s
demo-nginx-ingress-default-backend  0s

==> v1/ConfigMap
demo-nginx-ingress-controller  0s
==> v1/ServiceAccount
demo-nginx-ingress  0s
==> v1beta1/ClusterRole
demo-nginx-ingress  0s

==> v1beta1/RoleBinding
demo-nginx-ingress  0s

==> v1beta1/Deployment
demo-nginx-ingress-controller       0s
demo-nginx-ingress-default-backend  0s

==> v1/Pod(related)

NAME                                                READY  STATUS
RESTARTS  AGE
demo-nginx-ingress-controller-78cd47cf46-cw9q2      0/1    ContainerCreating
0         0s
demo-nginx-ingress-default-backend-5d47879fb7-5lptf  0/1    ContainerCreating
0         0s


NOTES:
The nginx-ingress controller has been installed.
It may take a few minutes for the LoadBalancer IP to be available.
You can watch the status by running 'kubectl --namespace demo get services -o
wide -w demo-nginx-ingress-controller'

An example Ingress that makes use of the controller:

  apiVersion: extensions/v1beta1
  kind: Ingress
  metadata:
    annotations:
      kubernetes.io/ingress.class: nginx
    name: example
    namespace: foo
  spec:
    rules:
      - host: www.example.com
        http:
          paths:
            - backend:
                serviceName: exampleService
                servicePort: 80
              path: /
    # This section is only required if TLS is to be enabled for the Ingress
    tls:
        - hosts:
            - www.example.com
          secretName: example-tls
```

```
If TLS is enabled for the Ingress, a Secret containing the certificate and key
must also be provided:

  apiVersion: v1
  kind: Secret
  metadata:
    name: example-tls
    namespace: foo
  data:
    tls.crt:<base64 encoded cert>
    tls.key:<base64 encoded key>
  type: kubernetes.io/tls
```

c. Verify the status of the NGINX Ingress Controller Helm deployment using the following command format:

```
kubectl get services <name of the NGINX Ingress Controller>
```

For example:

```
kubectl get services -o wide -w demo-nginx-ingress-controller --namespace demo
```

Example Output:

```
NAME    TYPE    CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE    SELECTOR
demo-nginx-ingress-controller   LoadBalancer   10.0.2.224   34.93.89.250
80:31294/TCP,443:32206/TCP   18d   app=nginx-
ingress,component=controller,release=demo-nginx-ingress
```

5. Install a single host simple fan-out Ingress resource Helm to route the traffic to the cluster-internal services.

   Note: The Ingress Resource determines the controller that is utilized to serve traffic. Set the Ingress annotation to select the NGINX ingress controller. This is set with an annotation, kubernetes.io/ingress.class, in the metadata section of the Ingress Resource. For example: annotations: kubernetes.io/ingress.class nginx

   The sample content of the `ingress.yaml` which is installed as a part of Ingress resource Helm Chart Installation is as follows.

   Observe the *kubernetes.io/ingress.class: nginx* annotation

```
kubectl create -f ../HelmCharts_xPlore/templates/xploreingress.yaml
```

   Sample:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
 name: xplore-ingress
 annotations:
 kubernetes.io/ingress.class: nginx
 nginx.ingress.kubernetes.io/affinity: cookie
 nginx.ingress.kubernetes.io/proxy-body-size: "0"
 nginx.ingress.kubernetes.io/proxy-connect-timeout: "1200"
 nginx.ingress.kubernetes.io/proxy-read-timeout: "1200"
 nginx.ingress.kubernetes.io/proxy-send-timeout: "1200"
 nginx.ingress.kubernetes.io/session-cookie-hash: sha1
 nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
 rules:
 #- host: {{ .Values.ingressHost | quote }}
 - http:
 paths:
```

```
- path: /dsearchadmin
backend:
serviceName: indexserver
servicePort: 9300
- path: /dsearch
backend:
serviceName: indexserver
servicePort: 9300
- path: /IndexAgent
backend:
serviceName: indexagent
servicePort: 9200
```

6. Perform the following to access the Documentum xPlore deployed inside the private cloud from outside:

   a. Obtain the external IP address of the load balancer service of the NGINX Ingress controller using the following command format:

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

      For example:

      ```
      kubectl get svc | grep demo-nginx-ingress
      ```

      Sample Output:

      ```
      NAME               TYPE           CLUSTER-IP      EXTERNAL-IP
      PORT(S)                      AGE
      demo-nginx-ingress-controller            LoadBalancer   10.0.2.224
      34.93.89.250     80:31294/TCP,443:32206/TCP   18d
      demo-nginx-ingress-default-backend       ClusterIP      10.0.6.231
      <none>           80/TCP                       18d
      ```

   b. Access the Documentum application using the external IP address of the NGINX Ingress controller load balancer service in a browser using the following URL format:

      ```
      http://<external IP address of NGINX Ingress controller load balancer service>/
      dsearchadmin
      ```

      ```
      http://<external IP address of NGINX Ingress controller load balancer service>/
      IndexAgent
      ```

5.3 Limitations

- External storage provisioners limitation. For deploying Documentum products, you need the capability to dynamically provision both ReadWriteOnce (RWO) Persistent Volumes (PVs) and ReadWriteMany (RWX) Persistent Volumes (PVs). Cloud providers provide the built-in provisioner to dynamically provision the ReadWriteOnce Persistent Volumes.
  For example, in Google Cloud Platform, if you specify the storageclass as standard in the Persistent volume Claim (PVC), then Google Cloud Platform automatically creates a Persistent Volume of requested size using Google Compute Engine Persistent Disk. However, the Google Compute Engine Persistent Disk does not support ReadWriteMany operation, and hence you have to provision a Google Cloud Filestore instance and external provisioner to dynamically manage the Persistent Volumes for ReadWriteMany Persistent Volume Claims.

- In order to achieve ingress in Google Cloud Platform or GKE Kubernetes cluster, Google Cloud Platform Google Cloud Load Balancer (GCLB) L7 Load balancer is not used as this Google Cloud Platform GCBL L7 load balancer does not communicate to services of the ClusterIP type on the backend. Use NGINX Ingress controller to achieve Ingress in a GKE cluster. Google Documentation contains more information about Ingress with NGINX controller on GKE.

5.4 Troubleshooting

There are no troubleshooting information for this release.

Chapter 6

Deploying Documentum xPlore on Amazon Web Services cloud platform

6.1 Prerequisites

Ensure that you complete the following activities before you deploy Documentum xPlore in the Amazon Web Services (AWS) environment:

1. Download and configure the Docker application from the Docker website.

   Docker Documentation contains detailed information.

2. Download and configure the Kubernetes application from the Kubernetes website.

   Kubernetes Documentation contains detailed information.

3. Download and install the latest version of Amazon Elastic Kubernetes Service (EKS), a managed Kubernetes service, to create a Kubernetes cluster on AWS.

4. You must be an Identity and Access Management (IAM) user with the following roles and permissions to create resources on AWS:

   - Roles: **EKS** to allow EKS to manage clusters and Elastic Compute Cloud (**EC2**) to allow instances to call AWS services.
   - Permissions: **elasticfilesystem:CreateFileSystem** and **elasticfilesystem:CreateMountTarget**.

5. Create an EKS cluster and a namespace within the cluster by performing the following steps:

   a. Install the AWS CLI on the client machine using the following command format:

   ```
   curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-
   bundle.zip"
   unzip awscli-bundle.zip
   sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws
   ```

   AWS Documentation contains more information.

   b. Configure your AWS CLI credentials using the following command format:

   ```
   [root@awsclient ~]# aws configure
   AWS Access Key ID [None]:  <your AWS Secret Access Key [None]:
   Default region name [None]:  <your AWS region>
   Default output format [None]:
   ```

   c. Install `eksctl` using the following command format:

   ```
   wget
   https://github.com/weaveworks/eksctl/releases/download/latest_release/
   eksctl_Linux_amd64.tar.gz
   ```

```
tar xz eksctl_Linux_amd64.tar.gz
sudo mv ./eksctl /usr/local/bin
```

d. Install the aws-iam-authenticator binary using the following command format:

```
wget https://amazon-eks.s3-us-west-2.amazonaws.com/1.14.6/2019-08-22/bin/linux/
amd64/aws-iam-authenticator
chmod +x ./aws-iam-authenticator
mv ./aws-iam-authenticator /usr/local/bin
```

e. Verify the `eksctl` installation using the following command format:

```
eksctl version
```

f. Install the `kubectl` binary using the following command:

```
curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/
1.14.6/2019-08-22/bin/linux/amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin
```

g. Verify the `kubectl` installation using the following command format:

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

h. Create an EKS Cluster mentioning the required cluster name, number of nodes, and your region using the following command format:

```
eksctl create cluster \
--name xploreeks \
--version 1.14 \
--region us-east-2 \
--nodegroup-name standard-workers \
--node-type m5.xlarge \
--nodes 3 \
--nodes-min 1 \
--nodes-max 4 \
--node-ami auto
```

This command may take a few minutes. Successful execution of this command creates an EKS cluster, nodes, default policies, and sets your `kubectl` configuration.

i. Create a Kubernetes cluster namespace on the Cloud shell using the following command format:

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

j. Verify if the namespace is created using the following command format:

```
kubectl get namespace
```

6. Upload the Docker images to Amazon Elastic Container Registry (ECR).

a. Authenticate ECR from CLI using the following command format:

```
C:\ aws ecr get-login --region < your aws region> --no-include-email
```

Copy and execute the output of this command in CLI. Successful execution of this command gives you authentication to ECR.

b. Create a repository in ECR to load your images by performing the following steps:

i. Open the Amazon ECR console.

   ii. From the navigation bar, select the region where you want to create your repository.

   iii. Click **Repositories**.

   iv. In the Repositories page, click **Create repository**.

   v. Enter an unique name for your repository to configure your repository.

   vi. For the image tag mutability, select the tag mutability setting for the repository. Repositories configured with immutable tags prevents image tags from being overwritten.

   AWS Documentation contains more information about the image tag mutability.

   vii. For scanning the image, select the image scanning setting for the repository. Repositories configured to scan on load, starts an image scan whenever an image is loaded. Otherwise, the image scans must be started manually.

   AWS Documentation contains more information about the scanning of image.

   viii. Click **Create repository**.

c. Tag your Docker images using the following command format:

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

d. Load the tagged image to ECR. For example:

```
docker push 707734542040.dkr.ecr.us-east-2.amazonaws.com/xplore/indexserver:
16.7.1000.0008
```

7. Verify the status of the deployed pods in the namespace using the following command format:

```
kubectl get pods -n <name of namespace>
```

Example output:

```
NAME READY STATUS RESTARTS AGE
xplore1cps-deploy-57c977bff7-vsvn6 1/1 Running 0 27h
```

8. Verify the version of Helm using the following command format:

```
helm version
```

Example output:

```
version.BuildInfo{Version:"v3.6.1",
GitCommit:"61d8e8c4a6f95540c15c6a65f36a6dd0a45e7a2f",
GitTreeState:"clean", GoVersion:"go1.16.5"}
```

9. Create an Amazon Elastic File System (EFS) file system by performing the following steps:

a. Obtain the cluster information: Before you create all the required resources to use Amazon EFS with your Amazon ECS cluster, obtain the basic information about the cluster, such as the VPC hosted inside, and the security group that it uses.

To obtain the VPC and security group IDs for a cluster: Open the Amazon EC2 console, select one of the container instances from your cluster and view the **Description** tab of the instance. Record the VPC ID value for your container instance. Then, create a security group and an Amazon EFS file system in this VPC. Open the security group to view its details and then record the Group ID. Then, allow the inbound traffic from this security group to your Amazon EFS file system.

b. Create a security group for an Amazon EFS file system: Create a security group for an Amazon EFS file system to allow inbound access from your container instances by performing the following steps:

   i. Open the Amazon EC2 console.
   ii. Click **Security Groups > Create Security Group**.
   iii. Enter an unique name for your security group. For example, `EFS-access-for-xploreeks-cluster`.
   iv. Enter a description for your security group.
   v. Select a VPC that you identified earlier for your cluster.
   vi. Click **Inbound > Add rule**.
   vii. Select **NFS** for type.
   viii. Select **Custom** for source.
   ix. Enter the security group ID that you identified earlier for your cluster.
   x. Click **Create**.

c. Create an Amazon EFS file system for Amazon ECS container instances by performing the following steps:

   i. Open the Amazon EFS console.
   ii. Click **Create file system**.
   iii. In the Configure file system access page, select the VPC where your container instances are hosted. By default, each subnet in the specified VPC receives a mount target that uses the default security group for that VPC.

      Note: Your Amazon EFS file system and your container instances must be in the same VPC.

   iv. Under **Create mount targets**, for **Security groups**, add the security group that you created.
   v. Click **Next step**.
   vi. Select a throughput mode for your file system.

      Note: By default, **Bursting** is the throughput mode and recommended for most file systems.

   vii. Select a performance mode for your file system.

      Note: By default, **General Purpose** is the performance mode and recommended for most file systems.

viii. Review your file system options and click **Create File System**.

10. Deploy an external storage provisioner (efs-provisioner) for dynamic provisioning of ReadWriteMany (RWX) Persistent Volumes (PVs) on Amazon EFS instance by performing the following steps:

    a. Download the external efs-provisioner Helm Chart from the Github website to your Cloud Shell machine.

    b. Install the efs-provisioner Helm Chart: Provide the EFS file system ID for efsProvisioner.efsFileSystemId and your AWS region for efsProvisioner.awsRegion. Successful deployment of this Helm gives a storageclass called efs with ReadWriteMany access.

    Example output:

```
helm install
--name efs-provisioner stable/efs-provisioner
--set efsProvisioner.efsFileSystemId=fs-12345678
--set efsProvisioner.awsRegion=us-east-2
--namespace demoDelete the PVC and ensure that the PV is also deleted.LAST
DEPLOYED: Tue Nov 26 17:44:25 2019
NAMESPACE: demo
STATUS: DEPLOYED

RESOURCES:
==> v1/ServiceAccount
NAME                 SECRETS  AGE
efs-provisioner  1              1s

==> v1/ClusterRole
NAME                AGE
efs-provisioner  1s

==> v1/ClusterRoleBinding
NAME                AGE
efs-provisioner  1s

==> v1/Deployment
NAME                DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
efs-provisioner  1              1                1
1                  1s

==> v1/Pod(related)
NAME                                              READY  STATUS
RESTARTS  AGE
efs-provisioner-5d4fc67f6d-4lc78  0/1        Init:0/1  0                  0s

==> v1beta1/StorageClass
NAME  PROVISIONER                 AGE
efs        example.com/aws-efs   1s
You can provision an EFS-based persistent volume with a persistent volume
claim as follows:
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my-efs-vol-1
  annotations:
    volume.beta.kubernetes.io/storage-class: efs
spec:
  storageClassName: efs
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Mi
```

11. Create a sample PVC (for example, `testPVC.yaml`) to test the dynamic provisioning of ReadWriteMany (RWX) PVs using the following command format:

    ```
    kind: PersistentVolumeClaim
    apiversion: v1
    metadata:
    name: test-claim
    spec:
    accessModes:
    - ReadWriteMany
    storageClassName: efs
    resources:
    requests:
    storage: 1Mi
    ```

12. Create the `testPVC.yaml` file using the following command format:

    ```
    C:\ kubectl create -f testPVC.yaml
    ```

    The following output is generated:

    ```
    persistentvolumeclaim/test-claim created
    ```

13. Verify if the corresponding Persistent Volume is created using the following command format:

    ```
    kubectl get pv
    ```

    Ensure the status of PV is `Bound`.

14. Delete the PVC and ensure that the PV is also deleted.

15. Install nginx-ingress Helm Chart for external access using the following command format:

    ```
    helm install –name <release name> stable/nginx-ingress --set rbac.create=true --
    namespace <name of namespace>
    ```

    For example:

    ```
    helm install --name demo-nginx-ingress stable/nginx-ingress --set rbac.create=true
    --namespace demo
    ```

    The nginx-ingress controller is installed. It may take a few minutes for the LoadBalancer IP to be available.

16. Download the Docker image from OpenText Container Registry. Perform the following tasks:

    a. Log in to OpenText Container Registry using the following command format:

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

       When prompted, provide your OpenText My Support login credentials.

    b. Download the Docker image using the following command format:

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

    The following Docker images are available for download:

Image name	Image tag
dctm_xplore_indexserver	22.1.0
dctm_xplore_indexagent	22.1.0
dctm_xplore_cps	22.1.0

17. Download the Helm charts from OpenText My Support.

18. Deploy Documentum Server as described in *OpenText Documentum Platform and Platform Extensions Cloud Deployment Guide*.

6.2 Deploying Documentum xPlore on AWS cloud platform

1. Extract the Helm Charts TAR file to a temporary location.

2. Upload the xPlore Docker images to Amazon Elastic Container Registry (ECR).

3. Install the Documentum xPlore Helm Charts by performing the following steps:

   a. Update the image and tag fields in the values.yaml files of your Helm Charts to point to the ECR images.

      For example:

```
### Docker Image ###
image:
  repository: 707734542040.dkr.ecr.us-east-2.amazonaws.com
  pullPolicy: IfNotPresent
  indexserver:
    name: xplore/indexserver
    tag: 16.7.1000.0008
  indexagent:
    name: xplore/indexagent
    tag: 16.7.1000.0008
  cps:
    name: xplore/cps
    tag: 16.7.1000.0008
```

   b. Update the storageClass fields in the values.yaml files with storageClass:efs for ReadWriteMany Persistent Volume Claim in the following format:

```
### Persistent Volume ###
volumeMounts:
  subPath: "."
persistentVolume:
  # possible value: true | false . isExist: true means using an external
volume instead of default one nested in xPlore chart
  isExist: false
  claimName: xpl-data-pvc
  # If isExist is true, then accessModes,size,storageClass will not be
effective.
  accessModes: ReadWriteMany
  size: 1Gi
  storageClass: efs
```

   c. xPlore Helm Chart installation:

i. Open the `xplore/values.yaml` file and provide the appropriate values for the variables depending on your environment to pass them to your templates. For descriptions of the parameters, see "Deploying xPlore on private cloud" on page 22.

ii. Deploy the Documentum xPlore Helm Chart in your AWS using the following command format:

```
helm install<location where Helm Chart Tar files are extracted>/xplore --
namespace <name of namespace> --name <release name>
```

For example:

```
helm install ../HelmCharts_xPlore/ --namespace demo --name xplore-aws
```

iii. Verify the status of the deployment of Documentum xPlore Helm Chart using the following command format:

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

Verify the status of the deployment of the Documentum xPlore pod using the following command format:

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

4. Install a single host simple fan-out Ingress resource Helm to route the traffic to the cluster-internal services.

Note: The Ingress Resource determines the controller that is utilized to serve traffic. Set the Ingress annotation to select the NGINX ingress controller. This is set with an annotation, `kubernetes.io/ingress.class`, in the metadata section of the Ingress Resource.

For example:

```
annotations: kubernetes.io/ingress.class nginx
```

The sample content of the `ingress.yaml` which is installed as a part of Ingress resource Helm Chart Installation is as follows.

Observe the `kubernetes.io/ingress.class: nginx` annotation

```
kubectl create -f ../HelmCharts_xPlore/templates/xploreingress.yaml
```

Sample:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: xplore-ingress
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  #- host: {{ .Values.ingressHost | quote }}
  - http:
      paths:
      - path: /dsearchadmin
        backend:
          serviceName: indexserver
          servicePort: 9300
      - path: /dsearch
        backend:
          serviceName: indexserver
        servicePort: 9300
    - path: /IndexAgent
      backend:
        serviceName: indexagent
        servicePort: 9200
```

5. Perform the following steps to access the Documentum xPlore deployed inside the private cloud from outside:

   a. Obtain the external IP address of the load balancer service of the NGINX Ingress controller using the following command format:

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

   For example:

   ```
   kubectl get svc | grep demo-nginx-ingress
   ```

   Sample Output:

   ```
   NAME                                           TYPE           CLUSTER-IP
   EXTERNAL-IP
   PORT(S)                                  AGE
   demo-nginx-ingress-controller        LoadBalancer   10.100.139.26
   a510cd7d40b7511eab94802b65a9671d-1402877647.us-east-2.elb.amazonaws.com
   80:30465/TCP,443:30282/TCP               14d
   demo-nginx-ingress-default-backend   ClusterIP      10.100.61.120   <none>
   80/TCP                                   14d
   ```

   b. Access the Documentum application using the external IP address of the NGINX Ingress controller load balancer service in a browser using the following URL format:
   `http://<external IP address of NGINX Ingress controller load balancer service>/dsearchadmin`
   `http://<external IP address of NGINX Ingress controller load balancer service>/IndexAgent`

   For example:
   `http://a510cd7d40b7511eab94802b65a9671d-1402877647.us-east-2.elb.amazonaws.com/dsearchadmin`
   `http://a510cd7d40b7511eab94802b65a9671d-1402877647.us-east-2.elb.amazonaws.com/IndexAgent`

6.3 Limitations

External storage provisioners limitation.

You need the capability to dynamically provision both ReadWriteOnce (RWO) Persistent Volumes (PVs) and ReadWriteMany (RWX) Persistent Volumes (PVs) to deploy Documentum products. Cloud providers provide the built-in provisioner to dynamically provision the ReadWriteOnce Persistent Volumes. For example, in AWS, if you specify the storageclass as `gp2` in the Persistent volume Claim (PVC), then AWS automatically creates a Persistent Volume of requested size using Amazon Elastic Book Store (EBS). However, the default (`gp2`) does not support ReadWriteMany operation, and hence you must provision an Amazon EFS file system and an external provisioner to dynamically manage the Persistent Volumes for ReadWriteMany Persistent Volume Claims.

6.4 Troubleshooting

There are no troubleshooting information for this release.

Chapter 7

Features for all cloud platforms

7.1 Integrating New Relic

New Relic is an application performance monitoring tool. This tool gives information that helps to analyze the application behavior, create real-time dashboards, and customization charts that are useful for application metrics.

7.1.1 Integrating New Relic with xPlore

The New Relic APM has been integrated in the container layer of xPlore for application monitoring. Please visit https://newrelic.com for more information about New Relic.

Open `xplore/values.yaml` file and provide the appropriate value for the variable depending on your environment to pass them to your templates as described in the following table:

Category	Name	Description
newRelic	enable	Set `true|false` to enable newRelic in the application. The default value is `true`. When set to `true`, the value of <*appNameSuffix*> and <*licenseKey*> must be set.
	appNameSuffix	Ensure the application name is unique. xPlore constructs the application name as <*NODE_NAME*>, <*appNameSuffix*> in `newrelic.yml`.
	licenseKey	Set the license key for newRelic.
	proxyHost	Set the proxy host if needed.
	proxyPort	Set the proxy port if needed.
	proxyScheme	Set the proxy scheme if needed.
	configMap	• <*isExist*>: Set as `true|false`. If set to `true`, using an existing configmap for new relic, then <*licenseKey*> and proxy details are not required. Instead, those properties must be included in the existing configmap file. The default value is `false`. • <*name*>: The name of the newRelic configmap. If it does not exist, the xPlore helm chart will create the newRelic configmap with this name. • <*key*>: The key name for the newRelic configuration in the configmap file. The default value is `newRelicConfigKey: newrelic.yml`.

7.1.2 Troubleshooting for New Relic integration

How to check the New Relic logs?

Run the command `kubectl exec -it <pod_name> bash -c <container_name>` to log in to the container.The logs are available in the `/opt/xPlore/newrelic/logs` folder.

Cannot find the applications in the New Relic website after starting the container.

- The New Relic SDK sends the data to the New Relic server. Verify if there are any network issues.
- Verify the proxy settings.

Chapter 8

Migrating Documentum xPlore to cloud platform

8.1 Migrating Documentum xPlore to clould environment

Migration from on-premises xPlore, user need keep data files and some key configuration files. Prepare the folders and configuration files which cloud xPlore could recognize them and copy these files to volume. Then deploy xPlore helm chart with this exsiting volume directly.

8.1.1 Prepare data folders and configuration files from on-premises xPlore

1. Create a folder named `rtdata`.
2. Create the following sub folders:
   - `config`
   - `data`
   - `wal`
3. Assume the instance name of the dsearch for an on-premises xPlore environment is `PrimaryDsearch`. Prepare each of the files from the on-premises xPlore environment as follows:
   a. `config:`copy all config files from `$XPLORE_HOME/config`
      i. update `indexserverconfig.xml`:
         - `index-server-configuration/node`:
           – hostname="indexserver-0"
           – url="http://indexserver:9300/dsearch";
         - `index-server-configuration/content-processing-services`:
           – set the local cps usage="search"
           – delete remote cps if it exists
         - `index-server-configuration/storage-locations/storage-location`:
           – change path="/opt/xPlore/rtdata/data"
      ii. update `XhiveDatabase.bootstrap`:

- server/node: host="indexserver-0"
- server/node/log: path="/opt/xPlore/rtdata/wal/PrimaryDsearch/PrimaryDsearch_wal"
- server/database: update all the file path from "C:/xPlore/data/..." to "/opt/xPlore/rtdata/data/..."

b. `data`: copy from `$XPLORE_HOME/data`.

c. `wal`: `mkdir ./PrimaryDsearch/PrimaryDsearch_wal copy` .wal files under `$XPLORE_HOME/config/wal/ to ./PrimaryDsearch/PrimaryDsearch_wal`

8.1.2 Deploy xPlore Helm chart with existing volume

1. Prepare a volume file in the cloud environment.

   For example, you can create a PVC named, `testpvc`, then deploy a Oracle Linux container which should load this PVC. Use the `kubectl cp` command to copy `rtdata` folder to the mount path. Now delete the Oracle Linux deployment and retain the PVC resource object.

2. Update `values.yaml` of xPlore helm chart.

   Update the indexserver and Persistent Volume part as follows:

```
update for indexserver:
  extraEnv:
    NODE_NAME: PrimaryDsearch
### Persistent Volume ###
volumeMounts:
  subPath: "rtdata"
persistentVolume:
  isExist: true
  claimName: testpvc
```

3. After all the pods status is ready, deploy xPlore Helm chart as described in step 5 on page 30 of “Deploying xPlore on private cloud” on page 22. Log in to dsearch admin web console, check if all the objects are listed.

4. Log in to IndexAgent interface and start the normal mode.

8.1.3 Limitations

- Any custom plug-in or configuration needs to be done through custom script by user after migrate to cloud xPlore.
- xPlore cloud environment currently supports only single node for Index Server and Index Agent instance.
- The new Index Agent register information may change in the cloud environment. It is recommended to delete the related objects in Content Server manually(`dm_ftengine_config`, `dm_ftindex_agent_config`, `dm_fulltext_index`) or uninstall the Index Agent in the old Content Server environment before migrating Content Server to cloud platform.