Cloud Pak for Data 4.0 Deployment Runbook

Air-gapped

Specialized installation approach

OCS

Hong Wei RJ Jia/China/IBM

Table of Contents

[Download server setup 4](#_Toc79489710)

[Set environment variables 4](#_Toc79489711)

[Downloading the Cloud Pak for Data cases 4](#_Toc79489712)

[Downloading the IBM Cloud Pak for Data platform operator package 4](#_Toc79489713)

[Configuring credentials for mirroring images 5](#_Toc79489714)

[Setting up an intermediary container registry 5](#_Toc79489715)

[Downloading IBM Cloud Pak foundational services 6](#_Toc79489716)

[Downloading service cases 6](#_Toc79489717)

[Mirroring the images to the intermediary registry 7](#_Toc79489718)

[Setting up a workstation to serve images 9](#_Toc79489719)

[Mirroring images to the private registry 9](#_Toc79489720)

[Configuring your cluster to pull images 12](#_Toc79489721)

[Configure global pull secret 12](#_Toc79489722)

[Configure image content policy 12](#_Toc79489723)

[Creating the catalog source 12](#_Toc79489724)

[Installing IBM Cloud Pak foundational services 14](#_Toc79489725)

[Creating operator subscriptions 16](#_Toc79489726)

[OperatorGroups 16](#_Toc79489727)

[Cloud Pak for Data Operator 16](#_Toc79489728)

[Namespace Scope Operator 17](#_Toc79489729)

[DMC Operator 17](#_Toc79489730)

[WSL Operator 17](#_Toc79489731)

[WML Operator 18](#_Toc79489732)

[Db2WH Operator 18](#_Toc79489733)

[WKC Operator 19](#_Toc79489734)

[Node Settings 19](#_Toc79489735)

[Install Cloud Pak for Data Instance1 21](#_Toc79489736)

[Install Zen service (control plane) 21](#_Toc79489737)

[Install WSL 22](#_Toc79489738)

[Install WML 23](#_Toc79489739)

[Install WKC 23](#_Toc79489740)

[Install Cloud Pak for Data Instance2 28](#_Toc79489741)

[Install Zen service 29](#_Toc79489742)

[Install Db2WH 30](#_Toc79489743)

[Install DMC 33](#_Toc79489744)

[Troubleshooting 34](#_Toc79489745)

[References 34](#_Toc79489746)

# Download server setup

Installing the software needed to mirror images

yum install openssl httpd-tools podman skopeo -y

mkdir -p /ibm

cd /ibm

CLOUDCTL

wget https://github.com/IBM/cloud-pak-cli/releases/download/v3.8.0/cloudctl-linux-amd64.tar.gz

tar -xf cloudctl-linux-amd64.tar.gz

cp cloudctl-linux-amd64 /usr/bin/cloudctl

OC CLIENT 4.6 or later

wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.6.31/openshift-client-linux-4.6.31.tar.gz

tar xvf openshift-client-linux-4.6.31.tar.gz

cp oc /usr/bin

cp kubectl /usr/bin

# Set environment variables

**Create a directory that serves as the offline store**

mkdir -p /data/offline

**Create the following environment variables with the installer image name and the image inventory on your host.**

export OFFLINEDIR=/data/offline

export CASE\_REPO\_PATH=https://github.com/IBM/cloud-pak/raw/master/repo/case

export CLOUDCTL\_TRACE=true # for extra logging

export USE\_SKOPEO=true

export ENTITLEMENT\_API\_KEY= *eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJJQk0gTWFya2V0cGxhY2UiLCJpYXQiOjE1OTkwMzg3NTksImp0aSI6IjQ5MjliNjYzYTA1NTQxNGZhNmY0Y2NjNjM2Y2IyZGEzIn0.ttXGplhS2elDBv66xbcyfPNsxAE6fM-\_IxDneuXds-E*

export OFFLINEDIR\_ARCHIVE=offline.tgz

export PORTABLE\_REGISTRY\_HOST=localhost

export PORTABLE\_REGISTRY\_PORT=5000

export PORTABLE\_REGISTRY=$PORTABLE\_REGISTRY\_HOST:$PORTABLE\_REGISTRY\_PORT

export PORTABLE\_REGISTRY\_USER=admin

export PORTABLE\_REGISTRY\_PASSWORD=passw0rd

export PORTABLE\_REGISTRY\_PATH=$OFFLINEDIR/imageregistry

# Downloading the Cloud Pak for Data cases

## Downloading the IBM Cloud Pak for Data platform operator package

cloudctl case save \

--case ${CASE\_REPO\_PATH}/ibm-cp-datacore-2.0.1.tgz \

--outputdir ${OFFLINEDIR} \

--no-dependency

## Configuring credentials for mirroring images

The IBM Cloud Pak CLI (cloudctl) includes an action called configure-cred-airgap. Run the appropriate commands to store the credentials that you will need to mirror images to the private container registry. The command stores the credentials to the following file on your local file system: $HOME/.airgap/secrets.

To configure the credentials that you need to mirror software images:

For cp.icr.io

Log in here: <https://myibm.ibm.com/products-services/containerlibrary>, using your intranet user and password. Then copy the entitlement key.

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--action configure-creds-airgap \

--args "--registry cp.icr.io --user cp --pass *eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJJQk0gTWFya2V0cGxhY2UiLCJpYXQiOjE1OTkwMzg3NTksImp0aSI6IjQ5MjliNjYzYTA1NTQxNGZhNmY0Y2NjNjM2Y2IyZGEzIn0.ttXGplhS2elDBv66xbcyfPNsxAE6fM-\_IxDneuXds-E* --inputDir ${OFFLINEDIR}"

Store authentication credentials of the portable container registry:

cloudctl case launch --case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz --inventory cpdPlatformOperator --action configure-creds-airgap --args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD}"

The command stores and caches the registry credentials in a file on your file system in the $HOME/.airgap/secrets location.

## Setting up an intermediary container registry

You must create an intermediary container registry where you can mirror the images before you can mirror them to the private registry.

To create an intermediary container registry:

1. To initialize the intermediary container registry, run the following command:

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--action init-registry \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --dir ${OFFLINEDIR}/imageregistry"

1. Set the following environment variable to ensure that the IBM Cloud Pak CLI (cloudctl) uses skopeo rather than oc mirror. The oc mirror command can have trouble pulling images from binary repositories.

export USE\_SKOPEO=true

1. To start the registry, run the following command:

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--action start-registry \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --dir ${OFFLINEDIR}/imageregistry"

1. To verify that the registry is running, run the following commands:

podman ps

# Downloading IBM Cloud Pak foundational services

Download this package if IBM Cloud Pak foundational services is not installed on the cluster.

cloudctl case save \

--case ${CASE\_REPO\_PATH}/ibm-cp-common-services-1.4.1.tgz \

--outputdir ${OFFLINEDIR}

# Downloading service cases

DMC

cloudctl case save \

--case ${CASE\_REPO\_PATH}/ibm-dmc-4.0.0.tgz \

--outputdir ${OFFLINEDIR}

DB2WH

cloudctl case save \

--case ${CASE\_REPO\_PATH}/ibm-db2wh-4.0.0.tgz \

--outputdir ${OFFLINEDIR}

WKC

cloudctl case save \

--case ${CASE\_REPO\_PATH}/ibm-wkc-4.0.0.tgz \

--outputdir ${OFFLINEDIR}

WML

cloudctl case save \

--case ${CASE\_REPO\_PATH}/ibm-wml-cpd-4.0.1.tgz \

--outputdir ${OFFLINEDIR}

WSL

cloudctl case save \

--case ${CASE\_REPO\_PATH}/ibm-wsl-2.0.0.tgz \

--outputdir ${OFFLINEDIR}

Jupyter Notebooks with Python 3.7 for GPU

cloudctl case save \

--case ${CASE\_REPO\_PATH}/ibm-wsl-runtimes-1.0.0.tgz \

--outputdir ${OFFLINEDIR}

Save the Docker registry image.

If your air-gapped network doesn’t have a Docker registry image, you can save the image and copy it later to the host in your air-gapped environment:

docker save docker.io/library/registry:2.7 -o $PORTABLE\_REGISTRY\_PATH/registry-image.tar

## Mirroring the images to the intermediary registry

To mirror the images:

1. Mirror the Cloud Pak for Data control plane images to the private container registry:

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --inputDir ${OFFLINEDIR}"

1. Mirror the images for each of the shared cluster components that you downloaded:

IBM Cloud Pak foundational services

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-common-services-1.4.1.tgz \

--inventory ibmCommonServiceOperatorSetup \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

1. Mirror the images for each of the services that you downloaded:

DMC

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-dmc-4.0.0.tgz \

--inventory dmcOperatorSetup \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

Db2WH

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-db2wh-4.0.0.tgz \

--inventory db2whOperatorSetup \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

WKC

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wkc-4.0.0.tgz \

--inventory wkcOperatorSetup \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

WSL

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wsl-2.0.0.tgz \

--inventory wslSetup \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

WML

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wml-cpd-4.0.1.tgz \

--inventory wmlOperatorSetup \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

## Setting up a workstation to serve images

Please refer to **Setting up a workstation to serve images** with this link https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=registry-mirroring-images-intermediary-container

## Mirroring images to the private registry

On the installation node of the OCP cluster.

1)Set the following environment variable to ensure that the IBM Cloud Pak CLI (cloudctl) uses skopeo rather than oc mirror. The oc mirror command can have trouble pulling images from binary repositories.

export USE\_SKOPEO=true

2)Set environment variables:

export OFFLINEDIR=/nfs/offline

export CLOUDCTL\_TRACE=true # for extra logging

export PORTABLE\_REGISTRY\_PATH=$OFFLINEDIR/imageregistry

export USE\_SKOPEO=true

export PORTABLE\_REGISTRY\_HOST=9.115.72.27

export PORTABLE\_REGISTRY\_PORT=5002

export PORTABLE\_REGISTRY=9.115.72.27:5002

export PORTABLE\_REGISTRY\_USER=admin

export PORTABLE\_REGISTRY\_PASSWORD=passw0rd

export PORTABLE\_REGISTRY\_PATH=$OFFLINEDIR/imageregistry

Here we take the Portable registry as the private registry

export PRIVATE\_REGISTRY\_USER=*admin*

export PRIVATE\_REGISTRY\_PASSWORD=*passw0rd*

export PRIVATE\_REGISTRY=9.115.72.27*:5002*

**Note:**

Use the ip address rather than localhost for the private registry.

3)On a portable storage device only, set up the registry. Run the local Docker registry as a container. The registry then points to the Docker file system directory that is transferred from the external host:

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--action init-registry \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --dir ${OFFLINEDIR}/imageregistry"

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--action start-registry \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --dir ${OFFLINEDIR}/imageregistry"

If the start-registry doesn’t work, then we can take this as the following workaround. And please kindly let me know if you encounter this problem. We may have to raise a defect.

podman run --name docker-registry-cpd -p ${PORTABLE\_REGISTRY\_PORT}:5000 --restart=always \

-v ${PORTABLE\_REGISTRY\_PATH}/data:/var/lib/registry:z \

-v ${PORTABLE\_REGISTRY\_PATH}/auth:/auth:z \

-v ${PORTABLE\_REGISTRY\_PATH}/certs:/certs:z \

-e REGISTRY\_AUTH=htpasswd \

-e REGISTRY\_AUTH\_HTPASSWD\_REALM=RegistryRealm \

-e REGISTRY\_AUTH\_HTPASSWD\_PATH=/auth/htpasswd \

-e REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/server.crt \

-e REGISTRY\_HTTP\_TLS\_KEY=/certs/server.key \

-d docker.io/library/registry:2.6

Verify

podman image search --list-tags 9.115.72.27:5002/zen-databases --authfile=/run/user/1000/containers/auth.json --tls-verify=false

The output should look like this:

[bjhwjia@bastion ~]$ podman image search --list-tags 9.115.72.27:5002/zen-databases --authfile=/run/user/1000/containers/auth.json --tls-verify=false

*NAME TAG*

*9.115.72.27:5002/zen-databases 4.0.0-1228-s390x*

*9.115.72.27:5002/zen-databases 4.0.0-1228-x86\_64*

*9.115.72.27:5002/zen-databases 4.0.0-1228-ppc64le*

4)Run the following command to store the credentials

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--action configure-creds-airgap \

--args "--registry ${PRIVATE\_REGISTRY} --user ${PRIVATE\_REGISTRY\_USER} --pass ${PRIVATE\_REGISTRY\_PASSWORD}"

The command stores and caches the registry credentials in a file on your file system in the $HOME/.airgap/secrets location.

5) Mirror the Cloud Pak for Data control plane images to the private container registry(no need to do this for Luban as we’ll use the portable registry as the private registry):

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--action mirror-images \

--args "--fromRegistry ${PORTABLE\_REGISTRY} --registry ${PRIVATE\_REGISTRY} --user ${PRIVATE\_REGISTRY\_USER} --pass ${PRIVATE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

6)Mirror the images for *each* of the shared cluster components that you downloaded:

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-common-services-1.4.0.tgz \

--inventory ibmCommonServiceOperatorSetup \

--action mirror-images \

--args "--fromRegistry ${PORTABLE\_REGISTRY} --registry ${PRIVATE\_REGISTRY} --user ${PRIVATE\_REGISTRY}\_USER --pass ${PRIVATE\_REGISTRY}\_PASSWORD --inputDir ${OFFLINEDIR}"

7)Mirror the images for each of the services that you downloaded.

Db2WH

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-db2wh-4.0.0.tgz \

--inventory db2whOperatorSetup \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

DMC

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-dmc-4.0.0.tgz \

--inventory dmcOperatorSetup \

--action mirror-images \

--args "--fromRegistry ${PORTABLE\_REGISTRY} --registry ${PRIVATE\_REGISTRY} --user ${PRIVATE\_REGISTRY}\_USER --pass ${PRIVATE\_REGISTRY}\_PASSWORD --inputDir ${OFFLINEDIR}"

WSL

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wsl-2.0.0.tgz \

--inventory wslSetup \

--action mirror-images \

--args "--fromRegistry ${PORTABLE\_REGISTRY} --registry ${PRIVATE\_REGISTRY} --user ${PRIVATE\_REGISTRY}\_USER --pass ${PRIVATE\_REGISTRY}\_PASSWORD --inputDir ${OFFLINEDIR}"

WML

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wml-cpd-4.0.0.tgz \

--inventory wmlOperatorSetup \

--action mirror-images \

--args "--fromRegistry ${PORTABLE\_REGISTRY} --registry ${PRIVATE\_REGISTRY} --user ${PRIVATE\_REGISTRY}\_USER --pass ${PRIVATE\_REGISTRY}\_PASSWORD --inputDir ${OFFLINEDIR}"

WKC

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wkc-4.0.0.tgz \

--inventory wkcOperatorSetup \

--action mirror-images \

--args "--registry ${PORTABLE\_REGISTRY} --user ${PORTABLE\_REGISTRY\_USER} --pass ${PORTABLE\_REGISTRY\_PASSWORD} --inputDir ${OFFLINEDIR}"

# Configuring your cluster to pull images

## Configure global pull secret

<https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=tasks-configuring-your-cluster-pull-images>

## Configure image content policy

[root@jhwcpd4-inf ~]# oc login https://api.jhwcpd4.cp.fyre.ibm.com:6443 -u kubeadmin -p ySamf-kTSZp-J9edB-SNZyt

cat <<EOF |oc apply -f -

apiVersion: operator.openshift.io/v1alpha1

kind: ImageContentSourcePolicy

metadata:

name: cloud-pak-for-data-mirror

spec:

repositoryDigestMirrors:

- mirrors:

- ${PRIVATE\_REGISTRY}/opencloudio

source: quay.io/opencloudio

- mirrors:

- ${PRIVATE\_REGISTRY}/cp

source: cp.icr.io/cp

- mirrors:

- ${PRIVATE\_REGISTRY}/cp/cpd

source: cp.icr.io/cp/cpd

- mirrors:

- ${PRIVATE\_REGISTRY}/cpopen

source: icr.io/cpopen

EOF

## Creating the catalog source

1.Create the IBM Cloud Pak foundational services catalog source. (Skip this step if you already have a catalog source for IBM Cloud Pak foundational services.)

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-common-services-1.4.1.tgz \

--inventory ibmCommonServiceOperatorSetup \

--namespace openshift-marketplace \

--action install-catalog \

--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"

2.Create the IBM Cloud Pak for Data catalog source

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-cp-datacore-2.0.1.tgz \

--inventory cpdPlatformOperator \

--namespace openshift-marketplace \

--action install-catalog \

--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"

3.Create the catalog source for each service that you mirrored to the private container registry.

DMC

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-dmc-4.0.0.tgz \

--inventory dmcOperatorSetup \

--namespace openshift-marketplace \

--action install-catalog \

--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"

DB2WH

yum install python2

yum install python2-pip

yum install -y python2

alternatives --set python /usr/bin/python2

pip2 install pyyaml

1)Db2U

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-db2uoperator-4.0.1.tgz \

--inventory db2uOperatorSetup \

--namespace openshift-marketplace \

--action install-catalog \

--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"

2)Db2WH

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-db2wh-4.0.0.tgz \

--inventory db2whOperatorSetup \

--namespace openshift-marketplace \

--action install-catalog \

--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"

WML

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wml-cpd-4.0.1.tgz \

--inventory wmlOperatorSetup \

--namespace openshift-marketplace \

--action install-catalog \

--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"

~~WML-A~~

~~cloudctl case launch \~~

~~--case ${OFFLINEDIR}/ibm-wml-accelerator-2.3.0.tgz \~~

~~--inventory wmla\_operator\_deploy \~~

~~--namespace openshift-marketplace \~~

~~--action install-catalog \~~

~~--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"~~

WSL

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wsl-2.0.0.tgz \

--inventory wslSetup \

--namespace openshift-marketplace \

--action install-catalog \

--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"

WKC

cloudctl case launch \

--case ${OFFLINEDIR}/ibm-wkc-4.0.0.tgz \

--inventory wkcOperatorSetup \

--namespace openshift-marketplace \

--action install-catalog \

--args "--registry ${PRIVATE\_REGISTRY} --inputDir ${OFFLINEDIR} --recursive"

## Installing IBM Cloud Pak foundational services

**1.Create the ibm-common-services project**

oc new-project ibm-common-services

**2.OperatorGroups**

*This installation will be single namespace for IBM Foundation. Please make sure only one operator group is defined in the namespace, OLM will give error, if you create two operator group in a single namespace*

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha2

kind: OperatorGroup

metadata:

name: operatorgroup

namespace: ibm-common-services

spec:

targetNamespaces:

- ibm-common-services

EOF

Status

oc get OperatorGroup operatorgroup -n ibm-common-services -o yaml

**3.Create the appropriate operator subscription for your environment:**

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

name: ibm-common-service-operator

namespace: ibm-common-services

spec:

channel: v3

installPlanApproval: Automatic

name: ibm-common-service-operator

source: opencloud-operators

sourceNamespace: openshift-marketplace

EOF

When you create the operator subscription:

* The IBM Cloud Pak foundational services operator installs the Operand Deployment Lifecycle Manager operator and the IBM NamespaceScope operator in the ibm-common-services project.
* The IBM Cloud Pak foundational services operator creates the CommonService custom resource.
* The Operand Deployment Lifecycle Manager operator creates the OperandRegistry, OperandConfig, and the OperatorBindInfo instances in the ibm-common-services project.

**Verifying the installation**

Verify the status of the operators by running the following commands:

1. Verify the status of ibm-common-service-operator:

oc --namespace *ibm-common-services* get csv

 If you installed IBM Cloud Pak foundational services in a different project, replace *ibm-common-services* with the correct project name.

The command should return output with the following format:

*NAME DISPLAY VERSION REPLACES PHASE*

*ibm-common-service-operator.v3.8 IBM Cloud Pak foundational services 3.8 ibm-common-service-operator.v3.7.0 Succeeded*

 Verify that the custom resource definitions were created:

oc get crd | grep operandrequest

 The command should return output with the following format:

NAME CREATED AT

operandrequests.operator.ibm.com 2021-06-23T10:10:22Z

 Confirm that IBM Cloud Pak foundational services API resources are available:

oc api-resources --api-group operator.ibm.com

# Creating operator subscriptions

Create the Cloud Pak for Data operator subscription

oc new-project cpd-operators

### OperatorGroups

*This installation will be single namespace for CPD operators. Please make sure only one operator group is defined in the namespace, OLM will give error, if you create two operator group in a single namespace*

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha2

kind: OperatorGroup

metadata:

name: operatorgroup

namespace: cpd-operators

spec:

targetNamespaces:

- cpd-operators

EOF

Status

oc get OperatorGroup operatorgroup -n cpd-operators -o yaml

### Cloud Pak for Data Operator

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

name: cpd-operator

  namespace: cpd-operators # Pick the project that contains the Cloud Pak for Data operator

spec:

channel: stable-v1

installPlanApproval: Automatic

name: cpd-platform-operator

source: cpd-platform

sourceNamespace: openshift-marketplace

EOF

### Namespace Scope Operator

If you are running a specialized installation (installing the IBM Cloud Pak for Data platform operator and the IBM Cloud Pak foundational services in separate projects), create an operator subscription for the IBM NamespaceScope Operator in the IBM Cloud Pak for Data platform operator project:

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

name: ibm-namespace-scope-operator

namespace: cpd-operators

spec:

channel: v3

installPlanApproval: Automatic

name: ibm-namespace-scope-operator

source: opencloud-operators

sourceNamespace: openshift-marketplace

EOF

Creating an operator subscription for services

### DMC Operator

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

  name: ibm-dmc-operator-subscription

  namespace: cpd-operators # Pick the project that contains the Cloud Pak for Data operator

spec:

  channel: v1.0

  installPlanApproval: Automatic

  name: ibm-dmc-operator

  source: ibm-dmc-operator-catalog

  sourceNamespace: openshift-marketplace

EOF

### WSL Operator

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

  annotations:

  name: ibm-cpd-ws-operator-catalog-subscription

  namespace: cpd-operators # Pick the project that contains the Cloud Pak for Data operator

spec:

  channel: v2.0

  installPlanApproval: Automatic

  name: ibm-cpd-wsl

  source: ibm-cpd-ws-operator-catalog

  sourceNamespace: openshift-marketplace

EOF

### WML Operator

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

  labels:

    app.kubernetes.io/instance: ibm-cpd-wml-operator-subscription

    app.kubernetes.io/managed-by: ibm-cpd-wml-operator

    app.kubernetes.io/name: ibm-cpd-wml-operator-subscription

  name: ibm-cpd-wml-operator-subscription

  namespace: cpd-operators # Pick the project that contains the Cloud Pak for Data operator

spec:

    channel: alpha

    installPlanApproval: Automatic

    name: ibm-cpd-wml-operator

    source: ibm-cpd-wml-operator-catalog

    sourceNamespace: openshift-marketplace

EOF

~~WML-A Operator~~

~~cat <<EOF |oc apply -f -~~

~~apiVersion: operators.coreos.com/v1alpha1~~

~~kind: Subscription~~

~~metadata:~~

~~name: ibm-cpd-wml-accelerator-operator~~

~~namespace: cpd-operators # Pick the project that contains the Cloud Pak for Data operator~~

~~spec:~~

~~name: ibm-cpd-wml-accelerator-operator~~

~~channel: WML-Accelerator-2.3~~

~~installPlanApproval: Automatic~~

~~source: ibm-cpd-wml-accelerator-catalog~~

~~sourceNamespace: openshift-marketplace~~

~~EOF~~

### Db2WH Operator

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

  name: ibm-db2uoperator-catalog-subscription

  namespace: cpd-operators # Pick the project that contains the Cloud Pak for Data operator

spec:

  channel: v1.1

  name: db2u-operator

  installPlanApproval: Automatic

  source: ibm-db2uoperator-catalog

  sourceNamespace: openshift-marketplace

EOF

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

  name: ibm-db2wh-cp4d-operator-catalog-subscription

  namespace: cpd-operators # Pick the project that contains the Cloud Pak for Data operator

spec:

  channel: v1.0

  name: ibm-db2wh-cp4d-operator

  installPlanApproval: Automatic

  source: ibm-db2wh-cp4d-operator-catalog

  sourceNamespace: openshift-marketplace

EOF

### WKC Operator

cat <<EOF |oc apply -f -

apiVersion: operators.coreos.com/v1alpha1

kind: Subscription

metadata:

  labels:

    app.kubernetes.io/instance:  ibm-cpd-wkc-operator-catalog-subscription

    app.kubernetes.io/managed-by: ibm-cpd-wkc-operator

    app.kubernetes.io/name:  ibm-cpd-wkc-operator-catalog-subscription

  name: ibm-cpd-wkc-operator-catalog-subscription

  namespace: cpd-operators # Pick the project that contains the Cloud Pak for Data operator

spec:

    channel: v1.0

    installPlanApproval: Automatic

    name: ibm-cpd-wkc

    source: ibm-cpd-wkc-operator-catalog

    sourceNamespace: openshift-marketplace

EOF

# Node Settings

##### This requires cluster administrator permission

<https://www.ibm.com/support/producthub/icpdata/docs/content/SSQNUZ_latest/cpd/install/node-settings.html>

There’s an known issue in the KC page with the above link for the CRIO settings.

https://github.ibm.com/PrivateCloud-analytics/Zen/issues/26870

For the time being, you can follow below for CRIO settings instead.

**CRIO settings**

Requires python3 for CRIO commands

default\_ulimits = [

"nofile=66560:66560" # recommended values

]

pids\_limit = 12288 # Recommended values

Check the current values

scp core@$(oc get nodes | grep worker | head -1 | awk '{print $1}'):/etc/crio/crio.conf /tmp/crio.conf

If current values are lower than recommended, please update the /tmp/crio.conf file and run the following commands

crio\_conf=$(cat /tmp/crio.conf | python3 -c "import sys, urllib.parse; print(urllib.parse.quote(''.join(sys.stdin.readlines())))")

Create machine config file 51-worker-cp4d-crio-conf.yaml file

cat << EOF > /tmp/51-worker-cp4d-crio-conf.yaml

apiVersion: machineconfiguration.openshift.io/v1

kind: MachineConfig

metadata:

labels:

machineconfiguration.openshift.io/role: worker

name: 51-worker-cp4d-crio-conf

spec:

config:

ignition:

version: 3.1.0

storage:

files:

- contents:

source: data:,${crio\_conf}

filesystem: root

mode: 0644

path: /etc/crio/crio.conf

EOF

Apply the new machineconfig to the cluster by running the following command:

oc create -f /tmp/51-worker-cp4d-crio-conf.yaml

# Install Cloud Pak for Data Instance1

We’ll take Specialized Installation approach.

Create a new project cpd-instance1

oc new-project cpd-instance1

Enable the IBM Cloud Pak for Data platform operator and the IBM Cloud Pak foundational services operator to watch the project where you will install IBM Cloud Pak for Data

Update the IBM NamespaceScope Operator in the Cloud Pak for Data operators project to watch the project where you plan to install Cloud Pak for Data.

Edit the namespaceMembers list to add the project where you plan to install Cloud Pak for Data. For example, if you plan to install Cloud Pak for Data in the cpd-instance project, add that project to the list:

cat <<EOF |oc apply -f -

apiVersion: operator.ibm.com/v1

kind: NamespaceScope

metadata:

name: cpd-operators

namespace: *cpd-operators* # (Default) Replace with the Cloud Pak for Data platform operator project name

spec:

namespaceMembers:

- *cpd-operators* # (Default) Replace with the Cloud Pak for Data platform operator project name

- cpd-instance1 # Replace with the project where you will install Cloud Pak for Data

EOF

## Install Zen service (control plane)

cat <<EOF |oc apply -f -

apiVersion: cpd.ibm.com/v1

kind: Ibmcpd

metadata:

name: *ibmcpd-cr* # This is the recommended name, but you can change it

namespace: cpd-instance1 # Replace with the project where you will install Cloud Pak for Data

spec:

license:

accept: true

license: *Enterprise* # Specify the Cloud Pak for Data license you purchased

storageClass: ocs-storagecluster-cephfs # Replace with the name of a RWX storage class

zenCoreMetadbStorageClass: *ocs-storagecluster-ceph-rbd* # (Recommended) Replace with the name of a RWO storage class

version: "4.0.1"

EOF

Get the status of the control plane (*ibmcpd-cr*):

oc get Ibmcpd ibmcpd-cr -o jsonpath="{.status.controlPlaneStatus}{'\n'}"

oc logs cpd-platform-operator-manager-d9597c56-8hbq6 -n cpd-operators -f

Get the status of the control plane (lite-cr):

oc get ZenService lite-cr -o jsonpath="{.status.zenStatus}{'\n'}"

The Cloud Pak for Data control plane is ready when the command returns Completed.

Get the URL of the Cloud Pak for Data web client:

oc get ZenService lite-cr -o jsonpath="{.status.url}{'\n'}"

Get the initial password for the admin user:

oc extract secret/admin-user-details --keys=initial\_admin\_password --to=-

**Troubleshooting**

oc logs $(oc get po -n cpd-operators| grep cpd-platform-operator-manager | awk '{print $1}') -n cpd-operators

oc get operandrequest

**Important**

Save the output of this command so that you can log in to the web client. It is strongly recommended that you change the initial password the first time that you log in to the web client.

## Install WSL

https://www.ibm.com/support/producthub/icpdata/docs/content/SSQNUZ\_latest/cpd/install/node-settings.html

cat <<EOF |oc apply -f -

apiVersion: ws.cpd.ibm.com/v1beta1

kind: WS

metadata:

name: ws-cr # This is the recommended name, but you can change it

namespace: cpd-instance1 # Replace with the project where you will install Watson Studio

spec:

docker\_registry\_prefix: cp.icr.io/cp/cpd

license:

accept: true

license: Enterprise # Specify the license you purchased

version: 4.0.0

storageVendor: ocs

storageClass: ocs-storagecluster-cephfs #if you use a different storage class, replace it with the appropriate storage class

EOF

**Note:**

If using nfs storage, no storageVendor section required.

Verifying the installation

oc get WS ws-cr -o jsonpath='{.status.wsStatus} {"\n"}'

Troubleshooting

oc logs $(oc get po -n cpd-operators| grep ibm-cpd-ws-operator | awk '{print $1}') -n cpd-operators -f

oc get operandrequest

## Install WML

https://www.ibm.com/support/producthub/icpdata/docs/content/SSQNUZ\_latest/svc-wml/wml-install.html

cat <<EOF |oc apply -f -

apiVersion: wml.cpd.ibm.com/v1beta1

kind: WmlBase

metadata:

name: wml-cr # This is the recommended name, but you can change it

namespace: cpd-instance1 # Replace with the project where you will install Watson Machine Learning

spec:

is\_35\_upgrade: false

ignoreForMaintenance: false

docker\_registry\_prefix: cp.icr.io/cp/cpd

license:

accept: true

license: Enterprise # Specify the license you purchased

version: 4.0.0

storageVendor: ocs

storageClass: ocs-storagecluster-cephfs #if you use a different storage class, replace it with the appropriate storage class

EOF

Verifying the installation

oc get WmlBase wml-cr -o jsonpath='{.status.wmlStatus} {"\n"}'

Troubleshooting

oc logs $(oc get po -n cpd-operators| grep ibm-cpd-wml-operator | awk '{print $1}') -n cpd-operators -f

oc get operandrequest

## Install WKC

Note: WKC doesn’t support the Specialized Installation approach.

This section here is just for demonstrating the installation process for WKC. Currently, it will fail with the WKC 4.0 version.

1.Creating custom security context constraints in Watson Knowledge Catalog

Define the SCC in the file wkc-iis-scc.yaml, as follows:

allowHostDirVolumePlugin: false

allowHostIPC: false

allowHostNetwork: false

allowHostPID: false

allowHostPorts: false

allowPrivilegeEscalation: true

allowPrivilegedContainer: false

allowedCapabilities: null

apiVersion: security.openshift.io/v1

defaultAddCapabilities: null

fsGroup:

type: RunAsAny

kind: SecurityContextConstraints

metadata:

annotations:

kubernetes.io/description: WKC/IIS provides all features of the restricted SCC

but runs as user 10032.

name: wkc-iis-scc

readOnlyRootFilesystem: false

requiredDropCapabilities:

- KILL

- MKNOD

- SETUID

- SETGID

runAsUser:

type: MustRunAs

uid: 10032

seLinuxContext:

type: MustRunAs

supplementalGroups:

type: RunAsAny

volumes:

- configMap

- downwardAPI

- emptyDir

- persistentVolumeClaim

- projected

- secret

users:

- system:serviceaccount:cpd-instance1:wkc-iis-sa

* Replace {{ namespace }} with the value for the actual namespace where Watson Knowledge Catalog is to be installed.
* If the custom SCC (wkc-iis-scc) already exists in the environment, delete the custom SCC that already exists and create a new custom SCC by using the YAML file from this step. Use the following command to delete the custom SCC: oc delete scc wkc-iis-scc

2)Run oc create to create the file:

oc create -f wkc-iis-scc.yaml

3) Run the following command to verify that the SCC was created:

$ oc get scc wkc-iis-scc

2.Installing Python

Watson Knowledge Catalog automatically installs Db2U and Db2® as a service. To ensure that these components can be installed, you must install the following Python software on the system where you issue the installation commands:

* Python 2
* pyyaml

To install Python 2, run the following command:

yum install -y python2

alternatives --set python /usr/bin/python2

To install pyyaml, run the following command:

pip install pyyaml

3.Installing the service

cat <<EOF |oc apply -f -

apiVersion: wkc.cpd.ibm.com/v1beta1

kind: WKC

metadata:

name: wkc-cr # This is the recommended name, but you can change it

namespace: cpd-instance1 # Replace with the project where you will install Watson Knowledge Catalog

spec:

license:

accept: true

license: Enterprise # Specify the license you purchased

version: 4.0.0

storageVendor: ocs # Specify the type of storage to use, such as ocs

# install\_wkc\_core\_only: true # To install the core version of the service, remove the comment tagging from the beginning of the line.

docker\_registry\_prefix: cp.icr.io/cp/cpd

useODLM: true

EOF

Verifying the installation

It might take two to three hours to install Watson Knowledge Catalog. You can check the status of Watson Knowledge Catalog by running the following command:

oc get WKC wkc-cr -o jsonpath='{.status.wkcStatus} {"\n"}'

You can check the status of the modules by running the following commands. The modules are installed in the order shown, and you must wait for each module to show Completed before the next module is installed.

To get the status of the common core services module, run the following command:

oc get CCS ccs-cr -o jsonpath='{.status.ccsStatus} {"\n"}'

To get the status of the Data Refinery module, run the following command:

oc get DataRefinery datarefinery-sample -o jsonpath='{.status.datarefineryStatus} {"\n"}'

To get the status of the Db2 as a service module, run the following command:

oc get Db2aaserviceService db2aaservice-cr -o jsonpath='{.status.db2aaserviceStatus} {"\n"}'

To get the status of the InfoSphere® Information Server module, run the following command. If you are installing the core version of the service, this check does not apply.

oc get IIS iis-cr -o jsonpath='{.status.iisStatus} {"\n"}'

To get the status of the Unified Governance module, run the following command. If you are installing the core version of the service, this check does not apply.

oc get UG ug-cr -o jsonpath='{.status.ugStatus} {"\n"}'

**Troubleshooting**

1)Check the logs of wkc operator

oc logs $(oc get po -n cpd-operators |grep ibm-cpd-wkc-operator | awk '{print $1}') -n cpd-operators -f

oc get operandrequest

2)[Db2U Operator is failing as it is looking for db2u-release ConfigMap in Bedrock namespace instead of Operators namespace](https://github.ibm.com/DB2/tracker/issues/11891)- Affect the scenario involving Bedrock/operators on different namespaces

Solution:

a) Change the META\_OPERATOR\_NAMESPACE from ibm-common-services to be cpd-operators

oc edit cm product-configmap

3) OperandRequest db2aaservice-db2uoperator-request stuck in pending

Text

Description automatically generated

Text

Description automatically generated

oc edit OperandRequest db2aaservice-db2uoperator-request

Text

Description automatically generated



4)WKC db2u operator stuck in 0/1 running status

[root@bastion ~]# oc get pods | grep wkc

c-db2oltp-wkc-db2u-0 0/1 Running 4 78m

[root@bastion ~]# oc describe pod c-db2oltp-wkc-db2u-0

level=error msg="exec failed: open /dev/tty: no such device or address"

Solution:

oc patch $( oc get sts -lapp=db2oltp-wkc -oname -n cpd-instance1) -n cpd-instance1 -p='{"spec":{"template":{"spec":{"containers":[{"name":"db2u","tty":false}]}}}}}'

5)IIS db2u operator stuck in 0/1 running status

[root@bastion ~]# oc get pods | grep -i iis

c-db2oltp-iis-db2u-0 0/1 Running 12 3h46m

[root@bastion ~]# oc describe pod c-db2oltp-iis-db2u-0

*Startup probe failed: time="2021-08-08T21:33:34+08:00" level=error msg="exec failed: open /dev/tty: no such device or address"*

Solution:

oc patch $( oc get sts -lapp=db2oltp-iis -oname -n cpd-instance1) -n cpd-instance1 -p='{"spec":{"template":{"spec":{"containers":[{"name":"db2u","tty":false}]}}}}}'

6) shop4info-schema-init job failed

[root@bastion ~]# oc get pods -A | grep -i shop4info-schema-init

*cpd-instance1 shop4info-schema-init-4td8p 0/1 Error 0 14m*

*cpd-instance1 shop4info-schema-init-528mh 0/1 Error 0 11m*

*cpd-instance1 shop4info-schema-init-dg58z 0/1 Error 0 14m*

*cpd-instance1 shop4info-schema-init-jkvxg 0/1 Error 0 8m17s*

*cpd-instance1 shop4info-schema-init-zdb7t 0/1 Error 0 13m*

Solution: Increase the *graph.properties.storage.connection-timeout* value

*# Please edit the object below. Lines beginning with a '#' will be ignored,*

*# and an empty file will abort the edit. If an error occurs while saving this file will be*

*# reopened with the relevant failures.*

*#*

*apiVersion: v1*

*data:*

*application.properties: |*

*# backend hosts*

*graph.properties.storage.hostname=cassandra*

*graph.properties.index.search.solr.zookeeper-url=zookeeper:2181/solr*

*graph.properties.storage.connection-timeout=960000*

# Install Cloud Pak for Data Instance2

Create a new project cpd-instance2

oc new-project cpd-instance2

Enable the IBM Cloud Pak for Data platform operator and the IBM Cloud Pak foundational services operator to watch the project where you will install IBM Cloud Pak for Data

Update the IBM NamespaceScope Operator in the Cloud Pak for Data operators project to watch the project where you plan to install Cloud Pak for Data.

Edit the namespaceMembers list to add the project where you plan to install Cloud Pak for Data. For example, if you plan to install Cloud Pak for Data in the cpd-instance project, add that project to the list:

cat <<EOF |oc apply -f -

apiVersion: operator.ibm.com/v1

kind: NamespaceScope

metadata:

name: cpd-operators

namespace: *cpd-operators* # (Default) Replace with the Cloud Pak for Data platform operator project name

spec:

namespaceMembers:

- *cpd-operators* # (Default) Replace with the Cloud Pak for Data platform operator project name

- cpd-instance1 # Replace with the project where you will install Cloud Pak for Data Instance1

- cpd-instance2 # Replace with the project where you will install Cloud Pak for Data Instance2

EOF

## Install Zen service

Create a custom resource to install Cloud Pak for Data in cpd-instance2.

cat <<EOF |oc apply -f -

apiVersion: cpd.ibm.com/v1

kind: Ibmcpd

metadata:

name: *ibmcpd-cr* # This is the recommended name, but you can change it

namespace: cpd-instance2 # Replace with the project where you will install Cloud Pak for Data

spec:

license:

accept: true

license: *Enterprise* # Specify the Cloud Pak for Data license you purchased

storageClass: ocs-storagecluster-cephfs # Replace with the name of a RWX storage class

zenCoreMetadbStorageClass: *ocs-storagecluster-ceph-rbd* # (Recommended) Replace with the name of a RWO storage class

version: "4.0.1"

EOF

Get the status of the control plane (*ibmcpd-cr*):

oc get Ibmcpd ibmcpd-cr -o jsonpath="{.status.controlPlaneStatus}{'\n'}"

oc logs cpd-platform-operator-manager-d9597c56-8hbq6 -n cpd-operators -f

Get the status of the control plane (lite-cr):

oc get ZenService lite-cr -o jsonpath="{.status.zenStatus}{'\n'}"

The Cloud Pak for Data control plane is ready when the command returns Completed.

Get the URL of the Cloud Pak for Data web client:

oc get ZenService lite-cr -o jsonpath="{.status.url}{'\n'}"

Get the initial password for the admin user:

oc extract secret/admin-user-details --keys=initial\_admin\_password --to=-

Troubleshooting

oc logs $(oc get po -n cpd-operators| grep cpd-platform-operator-manager | awk '{print $1}') -n cpd-operators

oc get operandrequest

**Important**

Save the output of this command so that you can log in to the web client. It is strongly recommended that you change the initial password the first time that you log in to the web client.

## Install Db2WH

1. Setting up dedicated nodes for your Db2 Warehouse deployment

When storage and compute share the same nodes, some additional considerations apply so that Db2 and software-defined storage can be properly scheduled on the same worker nodes.

If you are using OpenShift Container Storage, see [Managing container storage interface (CSI) component placements](https://www.ibm.com/links?url=https%3A%2F%2Faccess.redhat.com%2Fdocumentation%2Fen-us%2Fred_hat_openshift_container_storage%2F4.5%2Fhtml%2Fmanaging_openshift_container_storage%2Fmanaging-container-storage-interface-component-placements_rhocs" \o "(Opens in a new tab or window)" \t "_blank) for information about dedicating nodes that is specific to that platform.

1. Edit the configmap to add the toleration for the custom taint. Remember to save before exiting the editor.

*apiVersion: v1*

*data:*

*CSI\_LOG\_LEVEL: "5"*

*CSI\_PLUGIN\_TOLERATIONS: |2-*

*- key: node.ocs.openshift.io/storage*

*operator: Equal*

*value: "true"*

*effect: NoSchedule*

*- key : icp4data*

*operator: Equal*

*effect: NoSchedule*

*value: "database-db2wh"*

*CSI\_PROVISIONER\_TOLERATIONS: |2-*

*- key: node.ocs.openshift.io/storage*

*operator: Equal*

*value: "true"*

*effect: NoSchedule*

*- key : icp4data*

*operator: Equal*

*effect: NoSchedule*

*value: "database-db2wh"*

*kind: ConfigMap*

*metadata:*

*creationTimestamp: "2021-07-30T10:07:17Z"*

*name: rook-ceph-operator-config*

*namespace: openshift-storage*

*resourceVersion: "11578070"*

*selfLink: /api/v1/namespaces/openshift-storage/configmaps/rook-ceph-operator-config*

*uid: 04b6bfd9-a4e6-4527-8792-d8f3b73d1591*

1. Display the configmap to check the added toleration.

oc get configmap rook-ceph-operator-config -n openshift-storage -o yaml

**Verification step**

Verify that the csi-cephfsplugin-\* and csi-rbdplugin-\* pods are running on the dedicated nodes.

Verify that the rook-ceph-osd-x is schedulable on worker06

oc get pods -o wide -n openshift-storage

If the rook-ceph-osd-0-5f56d8b95c-vc9v9 scheduled on woker06 is in pending status, then run the following command:

oc edit pod rook-ceph-osd-0-5f56d8b95c-vc9v9 -n openshift-storage

Add toleration for the taint

tolerations:

………….

- effect: NoSchedule

key: node.kubernetes.io/memory-pressure

operator: Exists

- effect: NoSchedule

key: icp4data

operator: Equal

value: database-db2wh

1. Retrieve the name of the worker node that you want to dedicate to Db2 Warehouse:

oc get nodes

1. Taint the node with the NoSchedule effect and safely evict all of the pods from that node:

oc adm taint node *worker06* icp4data=*database-db2wh:*NoSchedule --overwrite

oc adm drain *worker06*

oc adm drain worker06 --ignore-daemonsets --delete-local-data --force

oc adm uncordon *worker06*

1. Label the node:

oc label node *worker06* icp4data=*database-db2wh* --overwrite

1. Verify that the node is labeled:

oc get node --show-labels

2.Install Db2wh

cat <<EOF |oc apply -f -

apiVersion: databases.cpd.ibm.com/v1

kind: Db2whService

metadata:

name: db2wh-cr # This is the recommended name, but you can change it

namespace: cpd-instance2 # Replace with the project where you will install Db2 Warehouse

spec:

license:

accept: true

license: Enterprise # Specify the license you purchased

EOF

**Verifying the installation**

oc get Db2whService db2wh-cr -o jsonpath='{.status.db2whStatus} {"\n"}'

3.Creating a database deployment

https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=warehouse-creating-database-deployment

ocs-storagecluster-cephfs should be used as the storage class.

**Troubleshooting:**

1)UI : Provision stuck in Step 2.

Backend: When describing the pod c-db2wh-1628147579678179-db2u-0, this error message is returned:

Readiness probe failed: time="2021-07-14T15:16:13Z" level=error msg="exec failed: open /dev/tty: no such device or address"

Solution:

oc patch $( oc get sts -lcomponent=db2wh -oname -n cpd-instance2) -n cpd-instance2 -p='{"spec":{"template":{"spec":{"containers":[{"name":"db2u","tty":false}]}}}}}'

2) [Db2U Operator is failing as it is looking for db2u-release ConfigMap in Bedrock namespace instead of Operators namespace](https://github.ibm.com/DB2/tracker/issues/11891)- Affect the scenario involving Bedrock/operators on different namespaces

oc edit cm grep product-configmap

Change the META\_OPERATOR\_NAMESPACE from ibm-common-services to be cpd-operators

## Install DMC

1)Install

Run the following commands to ensure that you can install the Redis operator

Run this command to support watching multiple namespaces using the NamespaceScope operator.

oc patch namespacescope cpd-operators --type='json' -p='[{"op":"replace", "path": "/spec/csvInjector/enable", "value":true}]' -n cpd-operators

cat <<EOF |oc apply -f -

apiVersion: dmc.databases.ibm.com/v1

kind: Dmcaddon

metadata:

name: dmc-addon # This is the recommended name, but you can change it

namespace: cpd-instance2 # Replace with the project where you will install Db2 Data Management Console

spec:

license:

accept: true

license: Enterprise # Specify the license you purchased

version: 4.0.0

EOF

Get the status of the dmc add-on :

oc get Dmcaddon dmc-addon -o jsonpath="{.status.dmcAddonStatus}{'\n'}"

**Troubleshooting**

oc logs $(oc get po -n cpd-operators| grep ibm-dmc-controller-manager | awk '{print $1}') -n cpd-operators

oc get operandrequest

2)Provision the service

<https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=console-provisioning-service>

Note: OpenShift Container Storage (ocs-storagecluster-cephfs) should be used as the storage class

Graphical user interface, application

Description automatically generated

# Troubleshooting

<https://ibm.box.com/s/chvmstcie0uoeaoxz3s48r3oq562urek>

# References

**Check the images in your image registry server**

*curl -k -u admin:passw0rd https://9.30.209.53:5000/v2/\_catalog?n=6000 | jq .repositories[]*

ls -R /data/offline

*/data/offline* is the Local repository file path of your image registry server

**Check your ImageContentSourcePolicy**

oc get imagecontentsourcepolicy cloud-pak-for-data-mirror -o yaml

**Create local RPM repo**

<https://linuxconfig.org/how-to-mount-umount-iso-image-on-centos-8-rhel-8-linux/>

https://access.redhat.com/solutions/3776721

mkdir /media/iso

mount -t iso9660 -o loop cpdauto\_artifacts\_rhel8/rhel-8.3-x86\_64-dvd.iso /media/iso

cd /etc/yum.repos.d

vi rhel8.repo

[RHEL\_8\_BaseOS]

name=dvdmedia

baseurl=file:///media/iso/BaseOS

enabled=1

gpgcheck=1

[RHEL\_8\_AppStream]

name=dvdmedia

baseurl=file:///media/iso/AppStream

enabled=1

gpgcheck=1

yum repolist

**Check Catalog Sources**

oc get CatalogSources -n openshift-marketplace

oc get pods -n openshift-marketplace