Red Hat OpenShift Container Storage 4.6

Deploying and managing OpenShift Container Storage using Red Hat Virtualization platform

How to install and manage

Last Updated: 2022-04-20

Red Hat OpenShift Container Storage 4.6 Deploying and managing OpenShift Container Storage using Red Hat Virtualization platform

How to install and manage

Legal Notice

Abstract

Read this document for instructions on installing and managing Red Hat OpenShift Container Storage using Red Hat Virtualization platform. Deploying and managing OpenShift Container Storage on Red Hat Virtualization platform is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Table of Contents

PREFACE

Red Hat OpenShift Container Storage 4.6 supports deployment on existing Red Hat OpenShift Container Platform (RHOCP) Red Hat Virtualization platform clusters.

To deploy OpenShift Container Storage in internal mode, follow the deployment process Deploying OpenShift Container Storage on Red Hat Virtualization.

CHAPTER 1. DEPLOYING OPENSHIFT CONTAINER STORAGE ON RED HAT VIRTUALIZATION PLATFORM

Deploying OpenShift Container Storage on OpenShift Container Platform using shared storage devices provided by Red Hat Virtualization installer-provisioned infrastructure (IPI) enables you to create internal cluster resources.

> **NOTE**
>
> Only internal Openshift Container Storage clusters are supported on Red Hat Virtualization platform. See Planning your deployment for more information about deployment requirements.

Use this section to deploy OpenShift Container Storage on Red Hat Virtualization infrastructure where OpenShift Container Platform is already installed.

To deploy Red Hat OpenShift Container Storage using local storage, follow these steps:

1. Understand the requirements for installing OpenShift Container Storage using local storage devices.
2. Install the Red Hat OpenShift Container Storage Operator .
3. Install Local Storage Operator .
4. Creating OpenShift Container Storage cluster service on Red Hat Virtualization .

1.1. REQUIREMENTS FOR INSTALLING OPENSHIFT CONTAINER STORAGE USING LOCAL STORAGE DEVICES

- You must upgrade to a latest version of OpenShift Container Platform 4.6 before deploying OpenShift Container Storage 4.6. For information, see Updating OpenShift Container Platform clusters guide.
- The Local Storage Operator version must match the Red Hat OpenShift Container Platform version in order to have the Local Storage Operator fully supported with Red Hat OpenShift Container Storage. The Local Storage Operator does not get upgraded when Red Hat OpenShift Container Platform is upgraded.
- You must have at least three OpenShift Container Platform worker nodes in the cluster with locally attached storage devices on each of them.
  - Each of the three selected nodes must have at least one raw block device available to be used by OpenShift Container Storage.
  - The devices you use must be empty; the disks must not include physical volumes (PVs), volume groups (VGs), or logical volumes (LVs) remaining on the disk.
- For minimum starting node requirements, see Resource requirements section in Planning guide.

1.2. INSTALLING RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR

You can install Red Hat OpenShift Container Storage Operator using the Red Hat OpenShift Container Platform Operator Hub. For information about the hardware and software requirements, see Planning your deployment.

Prerequisites

- You must be logged into the OpenShift Container Platform (RHOCP) cluster.
- You must have at least three worker nodes in the RHOCP cluster.

> **NOTE**
>
> - When you need to override the cluster-wide default node selector for OpenShift Container Storage, you can use the following command in command line interface to specify a blank node selector for the **openshift-storage** namespace:
>
> ```
> $ oc annotate namespace openshift-storage openshift.io/node-selector=
> ```
>
> - Taint a node as **infra** to ensure only Red Hat OpenShift Container Storage resources are scheduled on that node. This helps you save on subscription costs. For more information, see How to use dedicated worker nodes for Red Hat OpenShift Container Storage chapter in Managing and Allocating Storage Resources guide.

Procedure

1. Click **Operators → OperatorHub** in the left pane of the OpenShift Web Console.
2. Use **Filter by keyword** text box or the filter list to search for OpenShift Container Storage from the list of operators.
3. Click **OpenShift Container Storage**.
4. On the **OpenShift Container Storage operator** page, click Install.
5. On the **Install Operator** page, ensure the following options are selected by default::
   a. Update Channel as **stable-4.6**
   b. Installation Mode as **A specific namespace on the cluster**
   c. Installed Namespace as **Operator recommended namespace openshift-storage**. If Namespace **openshift-storage** does not exist, it will be created during the operator installation.
   d. Select **Enable operator recommended cluster monitoring on this namespace** checkbox as this is required for cluster monitoring.
   e. Select **Approval Strategy** as **Automatic** or **Manual**. Approval Strategy is set to **Automatic** by default.
      - **Approval Strategy** as **Automatic**.

> **NOTE**
>
> When you select the Approval Strategy as **Automatic**, approval is not required either during fresh installation or when updating to the latest version of OpenShift Container Storage.

i. Click **Install**

ii. Wait for the install to initiate. This may take up to 20 minutes.

iii. Click **Operators → Installed Operators**

iv. Ensure the **Project** is **openshift-storage**. By default, the **Project** is **openshift-storage**.

v. Wait for the **Status** of **OpenShift Container Storage** to change to **Succeeded**.

- **Approval Strategy** as **Manual**.

> **NOTE**
>
> When you select the Approval Strategy as **Manual**, approval is required during fresh installation or when updating to the latest version of OpenShift Container Storage.

i. Click **Install**

ii. On the **Manual approval required** page, you can either click **Approve** or **View Installed Operators in namespace openshift-storage** to install the operator.

> **IMPORTANT**
>
> Before you click either of the options, wait for a few minutes on the **Manual approval required** page until the install plan gets loaded in the window.

> **IMPORTANT**
>
> If you choose to click **Approve**, you must review the install plan before you proceed.

- If you click **Approve**.
  - Wait for a few minutes while the OpenShift Container Storage Operator is getting installed.
  - On the **Installed operator - ready for use** page, click **View Operator**.
  - Ensure the **Project** is **openshift-storage**. By default, the **Project** is **openshift-storage**.
  - Click **Operators → Installed Operators**
  - Wait for the **Status** of **OpenShift Container Storage** to change to **Succeeded**.

- If you click **View Installed Operators in namespace openshift-storage**.
  - On the **Installed Operators** page, click **ocs-operator**.
  - On the **Subscription Details** page, click the **Install Plan** link.
  - On the **InstallPlan Details** page, click **Preview Install Plan**.
  - Review the install plan and click **Approve**.
  - Wait for the **Status** of the **Components** to change from **Unknown** to either **Created** or **Present**.
  - Click **Operators → Installed Operators**
  - Ensure the **Project** is **openshift-storage**. By default, the **Project** is **openshift-storage**.
  - Wait for the **Status** of **OpenShift Container Storage** to change to **Succeeded**.

Verification steps

- Verify that **OpenShift Container Storage** Operator shows a green tick indicating successful installation.
- Click **View Installed Operators in namespace openshift-storage** link to verify that OpenShift Container Storage Operator shows the **Status** as **Succeeded** on the Installed Operators dashboard.

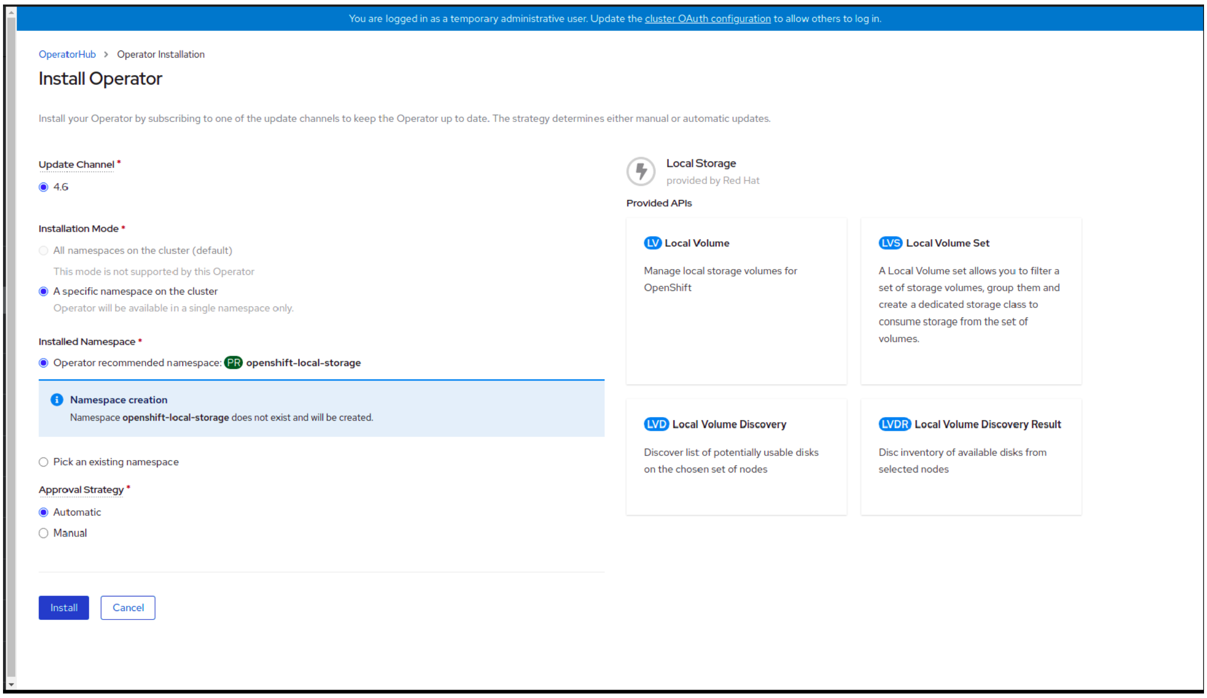
1.3. INSTALLING LOCAL STORAGE OPERATOR

Use this procedure to install the Local Storage Operator from the Operator Hub before creating OpenShift Container Storage clusters on local storage devices.

Procedure

1. Log in to the OpenShift Web Console.
2. Click **Operators → OperatorHub**.
3. Search for **Local Storage Operator** from the list of operators and click on it.
4. Click **Install**.

Figure 1.1. Install Operator page



5. Set the following options on the **Install Operator** page:
    a. Update Channel as **4.6**
    b. Installation Mode as **A specific namespace on the cluster**
    c. Installed Namespace as **Operator recommended namespace openshift-local-storage**.
    d. Approval Strategy as **Automatic**
6. Click **Install**.
7. Verify that the Local Storage Operator shows the Status as **Succeeded**.

1.4. CREATING OPENSHIFT CONTAINER STORAGE CLUSTER ON RED HAT VIRTUALIZATION PLATFORM

Use this procedure to create a storage cluster on Red Hat Virtualization platform when a storage class does not exist.

If you already have a storage class created, you can directly create a storage cluster as described in Creating a storage cluster on Red Hat Virtualization platform when a storage class exists .

Prerequisites

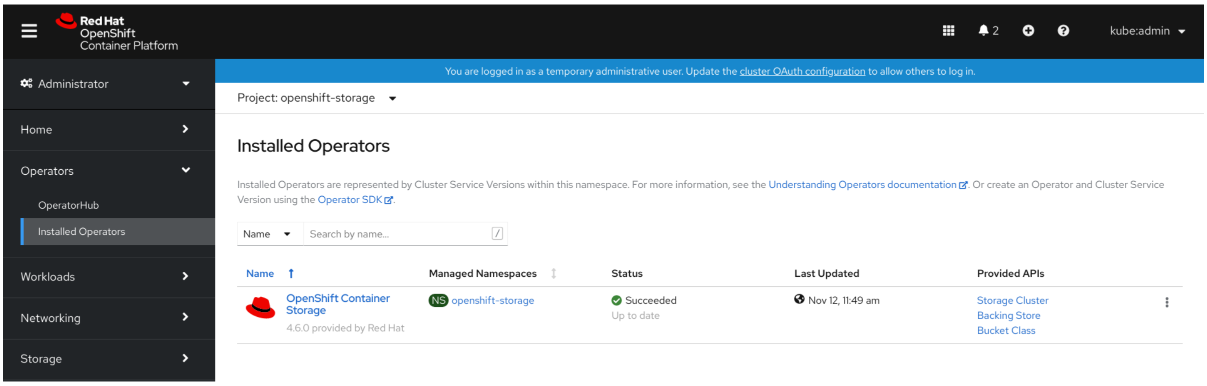
- Ensure that all the requirements in the Requirements for installing OpenShift Container Storage using local storage devices section are met.
- You must have a minimum of three worker nodes with the same storage type and size attached to each node to use local storage devices on Red Hat Virtualization platform.

Procedure

1. Log into the OpenShift Web Console.

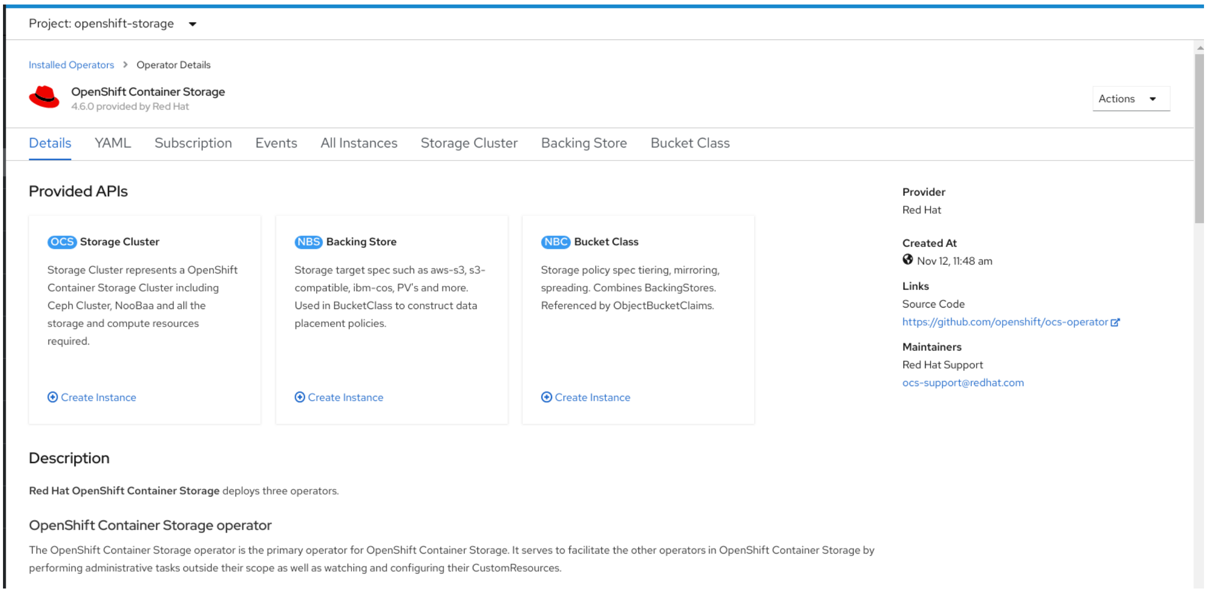
2. Click **Operators → Installed Operators** to view all the installed operators.
   Ensure that the **Project** selected is **openshift-storage**.

   Figure 1.2. OpenShift Container Storage Operator page



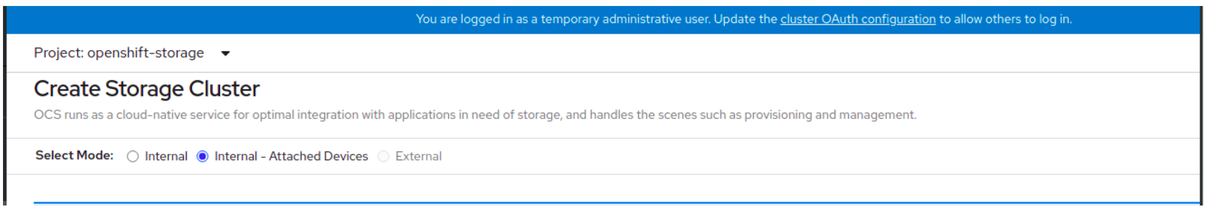
3. Click **OpenShift Container Storage**.

   Figure 1.3. Details tab of OpenShift Container Storage



4. Click **Create Instance** link of Storage Cluster.

   Figure 1.4. Create Storage Cluster page



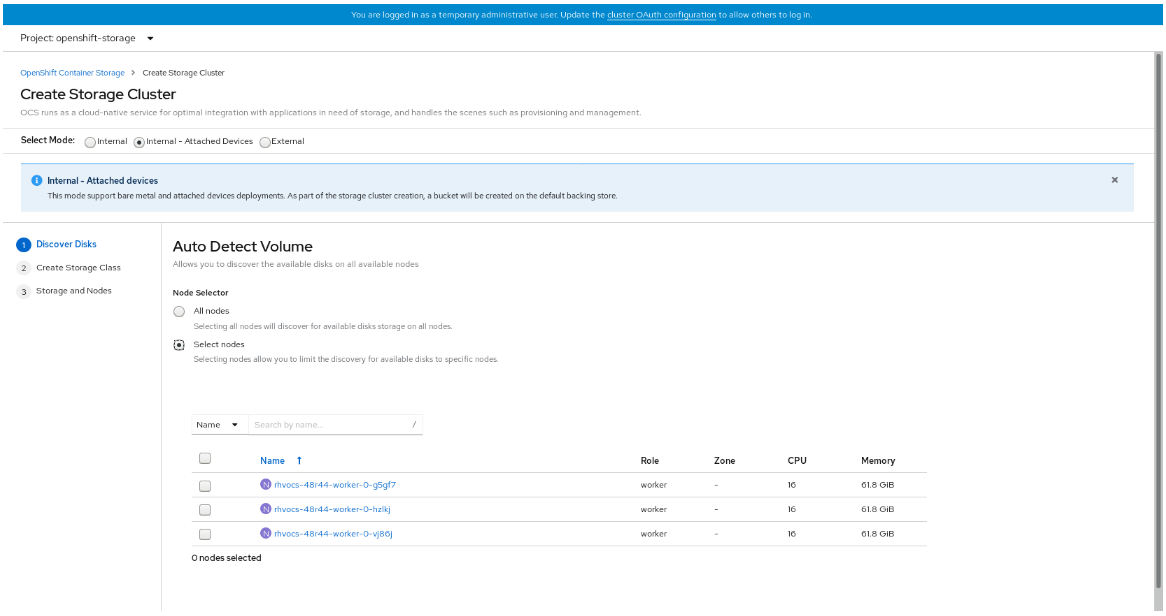
5. Select **Internal-Attached devices** for the **Select Mode**. By default, Internal is selected.

6. Create a storage cluster using the wizard that includes disk discovery, storage class creation, and storage cluster creation.
   You are prompted to install the Local Storage Operator. Click **Install** and install the operator as described in Installing Local Storage Operator.

   Discover Disks

You can discover a list of potentially usable disks on the selected nodes. Block disks and partitions that are not in use and available for provisioning persistent volumes (PVs) are discovered.

Figure 1.5. Discovery Disks wizard page



a. Choose one of the following:

- **All nodes** to discover disks in all the nodes.

- **Select nodes** to choose a subset of nodes from the nodes listed.
  To find specific worker nodes in the cluster, you can filter nodes on the basis of Name or Label. Name allows you to search by name of the node and Label allows you to search by selecting the predefined label.

  If the nodes selected do not match the OpenShift Container Storage cluster requirement of an aggregated 30 CPUs and 72 GiB of RAM, a minimal cluster will be deployed. For minimum starting node requirements, see Resource requirements section in Planning guide.

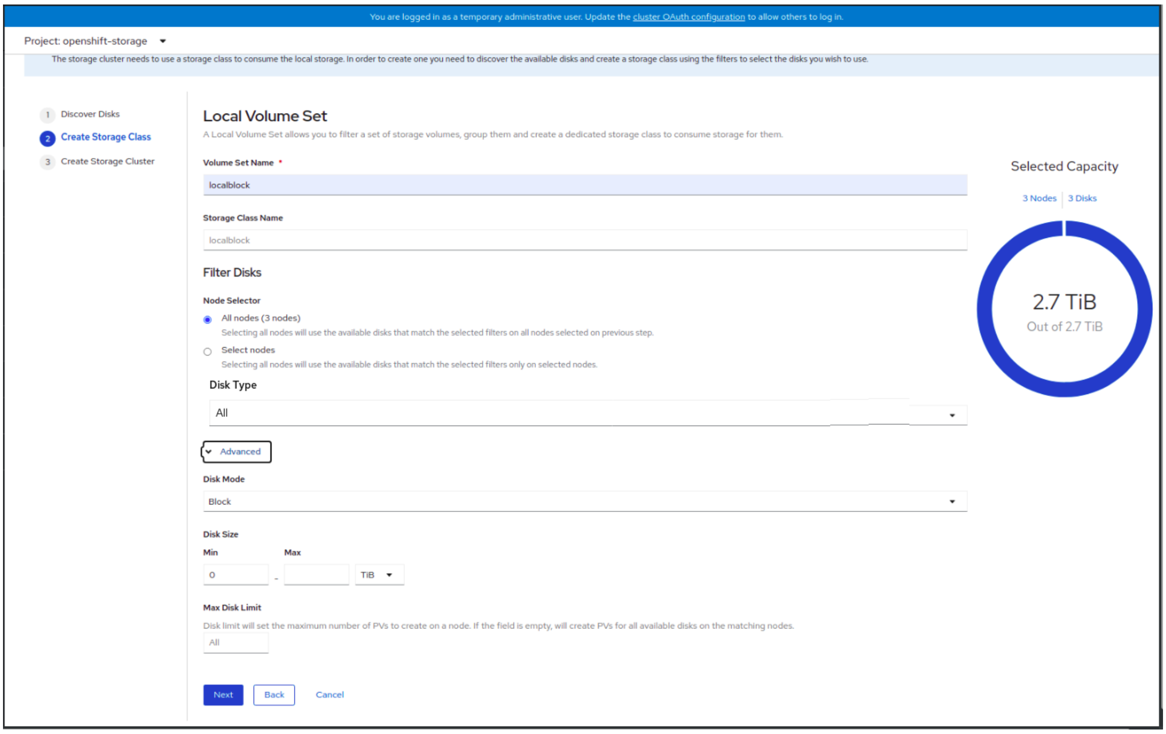
> **NOTE**
>
> If the nodes to be selected are tainted and not discovered in the wizard, follow the steps provided in the Red Hat Knowledgebase Solution as a workaround.

b. Click **Next**.

Create Storage Class

You can create a dedicated storage class to consume storage by filtering a set of storage volumes.

Figure 1.6. Create Storage Class wizard page



a. Enter the **Volume Set Name**.

b. Enter the **Storage Class Name**. By default, the volume set name appears for the storage class name.

c. The nodes selected for disk discovery in the earlier step are displayed in the **Filter Disks** section. Choose one of the following:

- **All nodes** to select all the nodes for which you discovered the devices.
- **Select nodes** to select a subset of the nodes for which you discovered the devices. To find specific worker nodes in the cluster, you can filter nodes on the basis of Name or Label. Name allows you to search by name of the node and Label allows you to search by selecting the predefined label.

  It is recommended that the worker nodes are spread across three different physical nodes, racks or failure domains for high availability.

  NOTE

  Ensure OpenShift Container Storage rack labels are aligned with physical racks in the datacenter to prevent a double node failure at the failure domain level.

d. Select the required **Disk Type**. The following options are available:

All	Selects all types of disks present on the nodes. By default, this option is selected.
SSD/NVME	Selects only SSD or NVME type of disks.

HDD	Selects only HDD type of disks.

NOTE

If the SSD/NVME disks are detected as HDD due to underlying abstraction of storage, select the disk type as **All** or **HDD**.

e. In the **Advanced** section, you can set the following:

Disk Mode	Block is selected by default.
Disk Size	Minimum and maximum avaialable size of the device that needs to be included. **NOTE** You must set a minimum size of 100GB for the device.
Max Disk Limit	This indicates the maximum number of PVs that can be created on a node. If this field is left empty, then PVs are created for all the available disks on the matching nodes.

f. (Optional) You can view the selected capacity of the disks on the selected nodes using the *Select Capacity* chart.
This chart might take a few minutes to reflect the disks that are discovered in the previous step.

You can click on the ***Nodes*** and ***Disks*** links on the chart to bring up the list of nodes and disks to view more details.

Figure 1.7. List of selected nodes

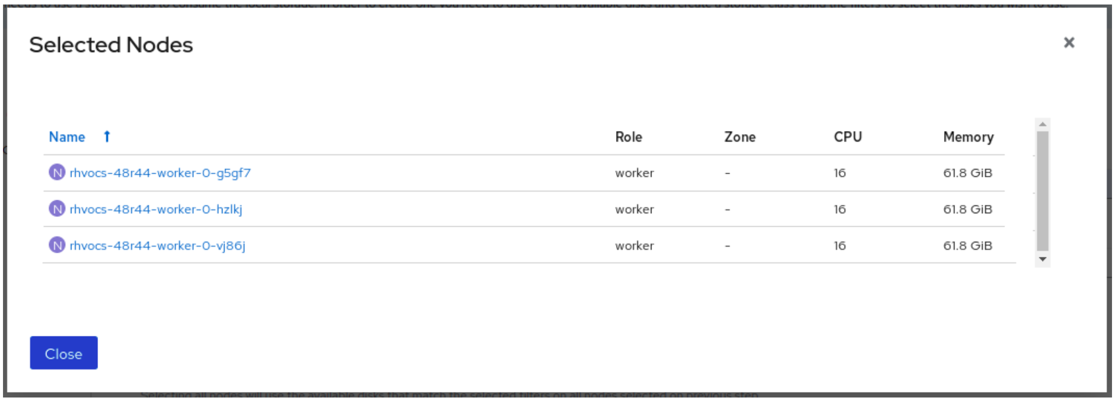
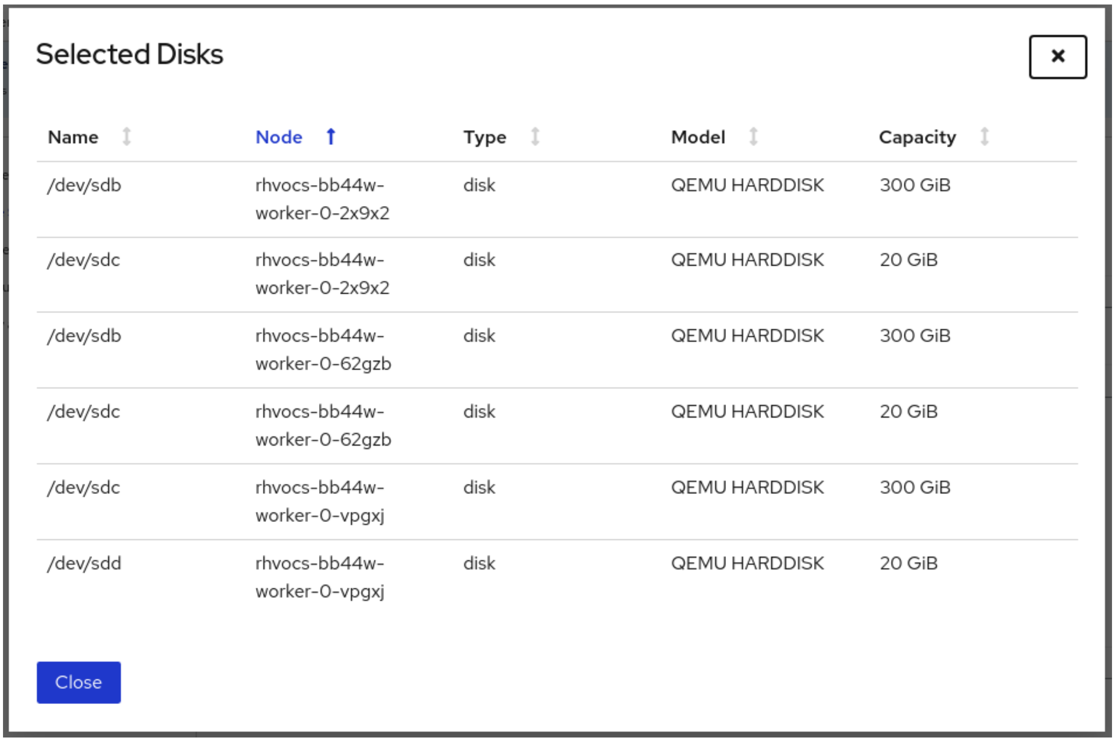


Figure 1.8. List of selected disks

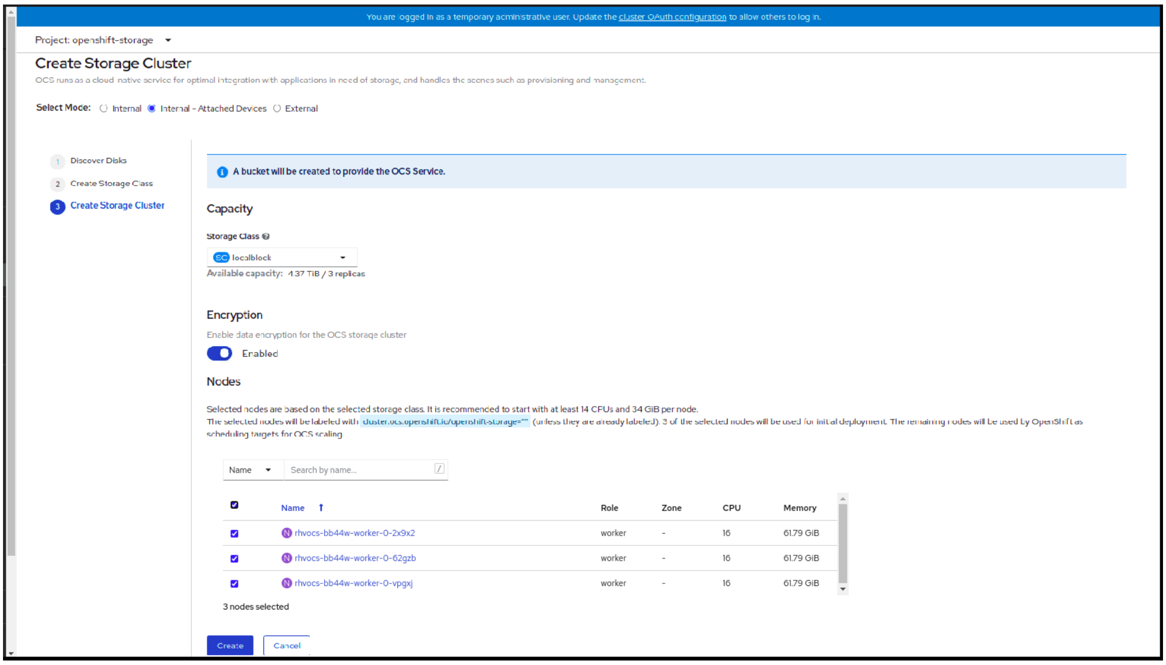


g. Click **Next**.

h. Click **Yes** in the message alert to confirm the creation of the storage class.
After the local volume set and storage class are created, it is not possible to go back to the step.

Create Storage Cluster

Figure 1.9. Create Storage Cluster wizard page



a. Select the required storage class.
You might need to wait a couple of minutes for the storage nodes corresponding to the selected storage class to get populated.

b. (Optional) In the Encryption section, set the toggle to Enabled to enable data encryption on the cluster.

c. The nodes corresponding to the storage class are displayed based on the storage class that you selected from the drop down list.

d. Click **Create**.
The **Create** button is enabled only when a minimum of three nodes are selected. A new storage cluster of three volumes will be created with one volume per worker node. The default configuration uses a replication factor of 3.

To expand the capacity of the initial cluster, see Scaling storage nodes .

Verification steps

See Verifying your OpenShift Container Storage installation .

1.5. CREATING A STORAGE CLUSTER ON RED HAT VIRTUALIZATION PLATFORM WHEN A STORAGE CLASS EXISTS

You can create a Openshift Container Storage Cluster using the existing storage class that is created through the Local Storage Operator page.

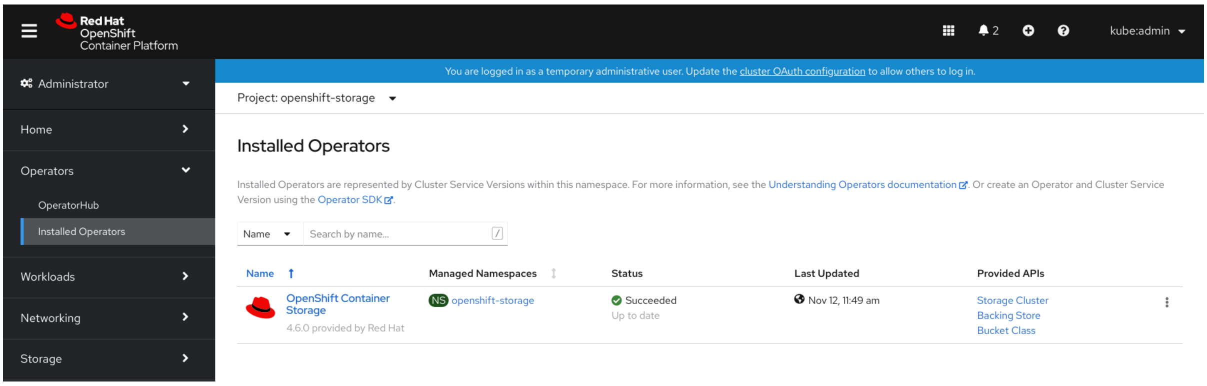
Prerequisites

- Ensure that all the requirements in the Requirements for installing OpenShift Container Storage using local storage devices section are met.
- You must have a minimum of three worker nodes with the same storage type and size attached to each node (for example, 2TB NVMe hard drive) to use local storage devices on bare metal.
- You must have created a storage class that consists of a minimum of three nodes and volume attached to it.

Procedure

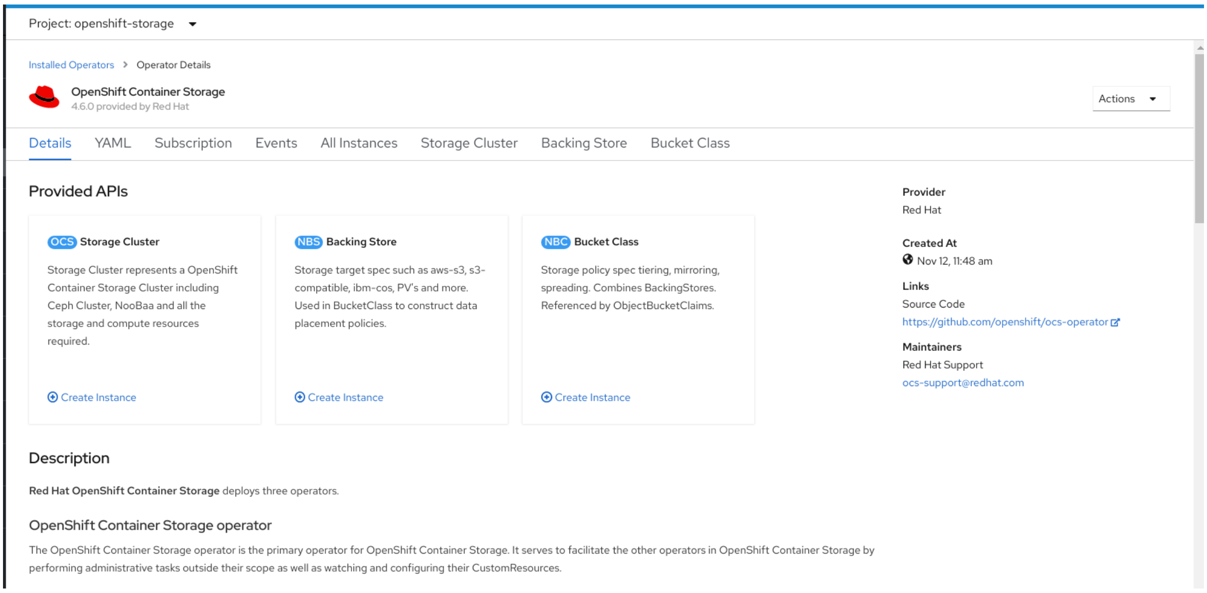
1. Log into the OpenShift Web Console.
2. Click **Operators → Installed Operators** to view all the installed operators.
Ensure that the **Project** selected is **openshift-storage**.

Figure 1.10. OpenShift Container Storage Operator page



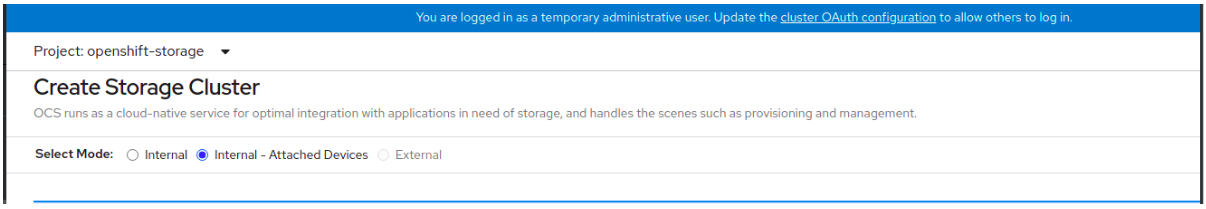
3. Click **OpenShift Container Storage**.

Figure 1.11. Details tab of OpenShift Container Storage



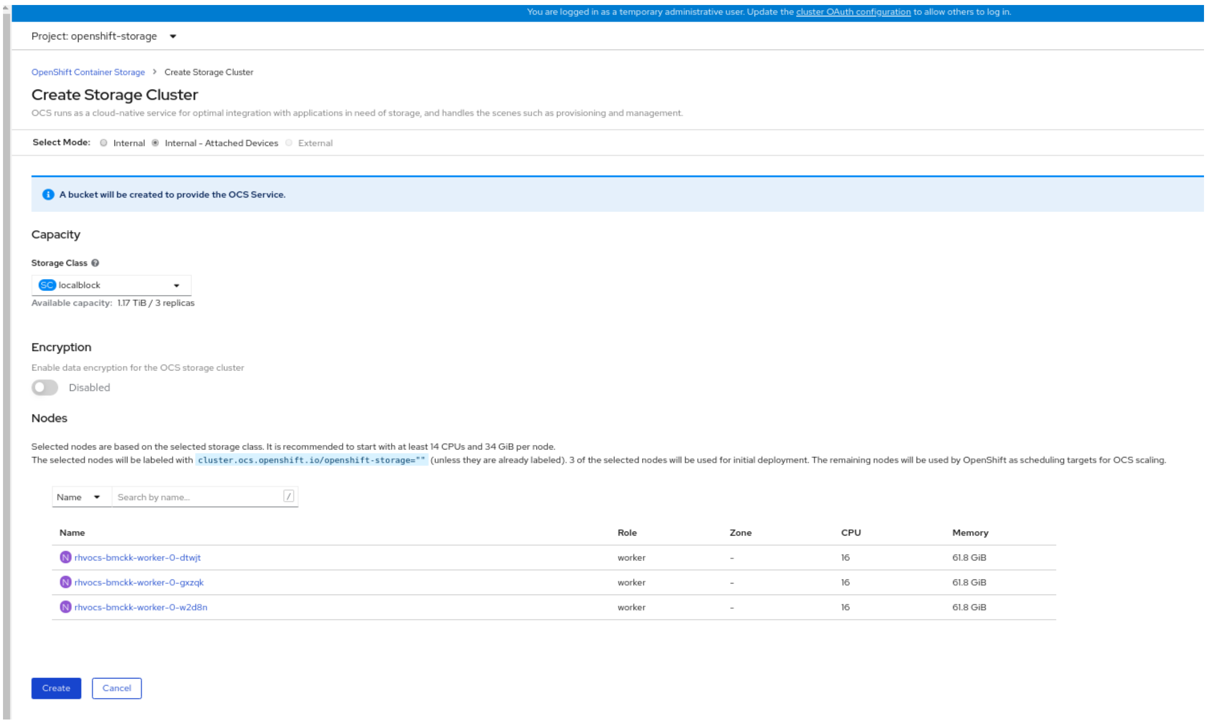
4. Click **Create Instance** link of Storage Cluster.

Figure 1.12. Create Storage Cluster page



5. Select **Internal-Attached devices** for the **Select Mode**. By default, Internal is selected.

Figure 1.13. Create Storage Cluster page



6. (Optional) In the Encryption section, set the toggle to Enabled to enable data encryption on the cluster.

7. The nodes corresponding to the selected storage class are displayed.
   The selected nodes are labeled with **cluster.ocs.openshift.io/openshift-storage=''** if they are not already labeled. Three of the selected nodes are used for initial deployment and the remaining nodes are used as the scheduling targets for OpenShift Container Storage scaling.

8. Click **Create**.
   The **Create** button is enabled only when a minimum of three nodes are selected.

   A new storage cluster of three volumes will be created with one volume per worker node. The default configuration uses a replication factor of 3.

To expand the capacity of the initial cluster, see Scaling storage nodes .

Verification steps

See Verifying your OpenShift Container Storage installation .

CHAPTER 2. VERIFYING OPENSHIFT CONTAINER STORAGE DEPLOYMENT

Use this section to verify that OpenShift Container Storage is deployed correctly.

2.1. VERIFYING THE STATE OF THE PODS

To determine if OpenShift Container storage is deployed successfully, you can verify that the pods are in **Running** state.

Procedure

1. Click **Workloads → Pods** from the left pane of the OpenShift Web Console.
2. Select **openshift-storage** from the **Project** drop down list.
   For more information on the expected number of pods for each component and how it varies depending on the number of nodes, see Table 2.1, "Pods corresponding to OpenShift Container storage cluster".
3. Verify that the following pods are in running and completed state by clicking on the **Running** and the **Completed** tabs:

Table 2.1. Pods corresponding to OpenShift Container storage cluster

Component	Corresponding pods
OpenShift Container Storage Operator	• **ocs-operator-*** (1 pod on any worker node) • **ocs-metrics-exporter-***
Rook-ceph Operator	**rook-ceph-operator-*** (1 pod on any worker node)
Multicloud Object Gateway	• **noobaa-operator-*** (1 pod on any worker node) • **noobaa-core-*** (1 pod on any storage node) • **noobaa-db-*** (1 pod on any storage node) • **noobaa-endpoint-*** (1 pod on any storage node)
MON	**rook-ceph-mon-*** (3 pods distributed across storage nodes)

Component	Corresponding pods
MGR	**rook-ceph-mgr-*** (1 pod on any storage node)
MDS	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** (2 pods distributed across storage nodes)
RGW	**rook-ceph-rgw-ocs-storagecluster-cephobjectstore-*** (2 pods distributed across storage nodes)

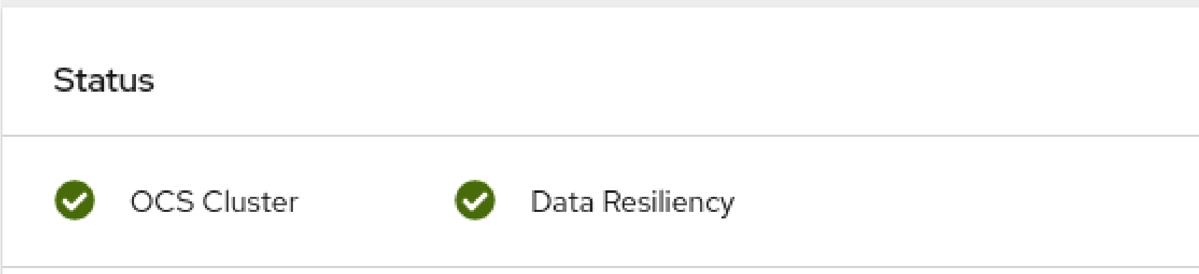
CSI	• **cephfs**
	  ◦ **csi-cephfsplugin-*** (1 pod on each worker node)
	  ◦ **csi-cephfsplugin-provisioner-*** (2 pods distributed across worker nodes)
	• **rbd**
	  ◦ **csi-rbdplugin-*** (1 pod on each worker node)
	  ◦ **csi-rbdplugin-provisioner-*** (2 pods distributed across worker nodes)

rook-ceph-crashcollector	**rook-ceph-crashcollector-*** (1 pod on each storage node)
OSD	• **rook-ceph-osd-*** (1 pod for each device) • **rook-ceph-osd-prepare-ocs-deviceset-*** (1 pod for each device)

2.2. VERIFYING THE OPENSHIFT CONTAINER STORAGE CLUSTER IS HEALTHY

- Click **Home → Overview** from the left pane of the OpenShift Web Console and click **Persistent Storage** tab.
- In the **Status card**, verify that *OCS Cluster* and *Data Resiliency* has a green tick mark as shown in the following image:

Figure 2.1. Health status card in Persistent Storage Overview Dashboard



- In the **Details card**, verify that the cluster information is displayed as follows:

  Service Name
  : OpenShift Container Storage

  Cluster Name
  : ocs-storagecluster-cephcluster

  Provider
  : oVirt

  Mode
  : Internal

  Version
  : ocs-operator-4.6.0

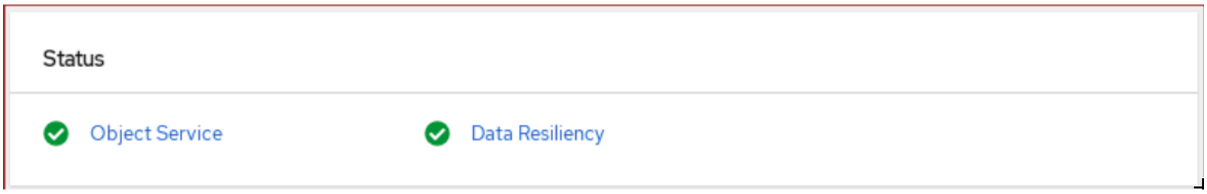
For more information on the health of OpenShift Container Storage cluster using the persistent storage dashboard, see Monitoring OpenShift Container Storage .

2.3. VERIFYING THE MULTICLOUD OBJECT GATEWAY IS HEALTHY

- Click **Home → Overview** from the left pane of the OpenShift Web Console and click the **Object Service** tab.

- In the **Status card**, verify that both *Object Service* and *Data Resiliency* are in **Ready** state (green tick).

Figure 2.2. Health status card in Object Service Overview Dashboard



- In the **Details card**, verify that the MCG information is displayed as follows:

  Service Name
  : OpenShift Container Storage

  System Name
  : Multicloud Object Gateway
  RADOS Object Gateway

  Provider

oVirt

Version

ocs-operator-4.6.0

For more information on the health of the OpenShift Container Storage cluster using the object service dashboard, see Monitoring OpenShift Container Storage .

2.4. VERIFYING THAT THE OPENSHIFT CONTAINER STORAGE SPECIFIC STORAGE CLASSES EXIST

To verify the storage classes exists in the cluster:

- Click **Storage → Storage Classes** from the left pane of the OpenShift Web Console.
- Verify that the following storage classes are created with the OpenShift Container Storage cluster creation:
  - **ocs-storagecluster-ceph-rbd**
  - **ocs-storagecluster-cephfs**
  - **openshift-storage.noobaa.io**
  - **ocs-storagecluster-ceph-rgw**

CHAPTER 3. UNINSTALLING OPENSHIFT CONTAINER STORAGE

3.1. UNINSTALLING OPENSHIFT CONTAINER STORAGE IN INTERNAL MODE

Use the steps in this section to uninstall OpenShift Container Storage.

Uninstall Annotations

Annotations on the Storage Cluster are used to change the behavior of the uninstall process. To define the uninstall behavior, the following two annotations have been introduced in the storage cluster:

- **uninstall.ocs.openshift.io/cleanup-policy: delete**
- **uninstall.ocs.openshift.io/mode: graceful**

The below table provides information on the different values that can used with these annotations:

Table 3.1. uninstall.ocs.openshift.io uninstall annotations descriptions

Annotation	Value	Default	Behavior
cleanup-policy	delete	Yes	Rook cleans up the physical drives and the **DataDirHostPath**
cleanup-policy	retain	No	Rook does **not** clean up the physical drives and the **DataDirHostPath**
mode	graceful	Yes	Rook and NooBaa **pauses** the uninstall process until the PVCs and the OBCs are removed by the administrator/user
mode	forced	No	Rook and NooBaa proceeds with uninstall even if PVCs/OBCs provisioned using Rook and NooBaa exist respectively.

You can change the cleanup policy or the uninstall mode by editing the value of the annotation by using the following commands:

```
$ oc annotate storagecluster -n openshift-storage ocs-storagecluster
uninstall.ocs.openshift.io/cleanup-policy="retain" --overwrite
storagecluster.ocs.openshift.io/ocs-storagecluster annotated
```

```
$ oc annotate storagecluster -n openshift-storage ocs-storagecluster
uninstall.ocs.openshift.io/mode="forced" --overwrite
storagecluster.ocs.openshift.io/ocs-storagecluster annotated
```

Prerequisites

- Ensure that the OpenShift Container Storage cluster is in a healthy state. The uninstall process can fail when some of the pods are not terminated successfully due to insufficient resources or nodes. In case the cluster is in an unhealthy state, contact Red Hat Customer Support before uninstalling OpenShift Container Storage.
- Ensure that applications are not consuming persistent volume claims (PVCs) or object bucket claims (OBCs) using the storage classes provided by OpenShift Container Storage.
- If any custom resources (such as custom storage classes, cephblockpools) were created by the admin, they must be deleted by the admin after removing the resources which consumed them.

Procedure

1. Delete the volume snapshots that are using OpenShift Container Storage.
   a. List the volume snapshots from all the namespaces.

      ```
      $ oc get volumesnapshot --all-namespaces
      ```

   b. From the output of the previous command, identify and delete the volume snapshots that are using OpenShift Container Storage.

      ```
      $ oc delete volumesnapshot <VOLUME-SNAPSHOT-NAME> -n <NAMESPACE>
      ```

2. Delete PVCs and OBCs that are using OpenShift Container Storage.
   In the default uninstall mode (graceful), the uninstaller waits till all the PVCs and OBCs that use OpenShift Container Storage are deleted.

   If you wish to delete the Storage Cluster without deleting the PVCs beforehand, you may set the uninstall mode annotation to "forced" and skip this step. Doing so will result in orphan PVCs and OBCs in the system.

   a. Delete OpenShift Container Platform monitoring stack PVCs using OpenShift Container Storage.
      See Section 3.2, “Removing monitoring stack from OpenShift Container Storage”
   b. Delete OpenShift Container Platform Registry PVCs using OpenShift Container Storage.
      See Section 3.3, “Removing OpenShift Container Platform registry from OpenShift Container Storage”
   c. Delete OpenShift Container Platform logging PVCs using OpenShift Container Storage.
      See Section 3.4, “Removing the cluster logging operator from OpenShift Container Storage”
   d. Delete other PVCs and OBCs provisioned using OpenShift Container Storage.
      - Given below is a sample script to identify the PVCs and OBCs provisioned using OpenShift Container Storage. The script ignores the PVCs that are used internally by Openshift Container Storage.

```
#!/bin/bash

RBD_PROVISIONER="openshift-storage.rbd.csi.ceph.com"
CEPHFS_PROVISIONER="openshift-storage.cephfs.csi.ceph.com"
NOOBAA_PROVISIONER="openshift-storage.noobaa.io/obc"
RGW_PROVISIONER="openshift-storage.ceph.rook.io/bucket"

NOOBAA_DB_PVC="noobaa-db"
NOOBAA_BACKINGSTORE_PVC="noobaa-default-backing-store-noobaa-pvc"

# Find all the OCS StorageClasses
OCS_STORAGECLASSES=$(oc get storageclasses | grep -e
"$RBD_PROVISIONER" -e "$CEPHFS_PROVISIONER" -e
"$NOOBAA_PROVISIONER" -e "$RGW_PROVISIONER" | awk '{print $1}')

# List PVCs in each of the StorageClasses
for SC in $OCS_STORAGECLASSES
do
        echo
"==============================================================================
=="
        echo "$SC StorageClass PVCs and OBCs"
        echo
"==============================================================================
=="
        oc get pvc  --all-namespaces --no-headers 2>/dev/null | grep $SC | grep -v -e
"$NOOBAA_DB_PVC" -e "$NOOBAA_BACKINGSTORE_PVC"
        oc get obc  --all-namespaces --no-headers 2>/dev/null | grep $SC
        echo
done
```

NOTE

Omit **RGW_PROVISIONER** for cloud platforms.

- Delete the OBCs.

```
$ oc delete obc <obc name> -n <project name>
```

- Delete the PVCs.

```
$ oc delete pvc <pvc name> -n <project-name>
```

NOTE

Ensure that you have removed any custom backing stores, bucket classes, etc., created in the cluster.

3. Delete the Storage Cluster object and wait for the removal of the associated resources.

```
$ oc delete -n openshift-storage storagecluster --all --wait=true
```

4. Check for cleanup pods if the **uninstall.ocs.openshift.io/cleanup-policy** was set to **delete**(default) and ensure that their status is **Completed**.

```
$ oc get pods -n openshift-storage | grep -i cleanup
NAME                                READY   STATUS      RESTARTS   AGE
cluster-cleanup-job-<xx>        0/1     Completed   0          8m35s
cluster-cleanup-job-<yy>      0/1     Completed   0          8m35s
cluster-cleanup-job-<zz>      0/1     Completed   0          8m35s
```

5. Confirm that the directory **/var/lib/rook** is now empty. This directory will be empty only if the **uninstall.ocs.openshift.io/cleanup-policy** annotation was set to **delete**(default).

```
$ for i in $(oc get node -l cluster.ocs.openshift.io/openshift-storage= -o jsonpath='{
.items[*].metadata.name }'); do oc debug node/${i} -- chroot /host  ls -l /var/lib/rook; done
```

6. If encryption was enabled at the time of install, remove **dm-crypt** managed **device-mapper** mapping from OSD devices on all the OpenShift Container Storage nodes.

   a. Create a **debug** pod and **chroot** to the host on the storage node.

   ```
   $ oc debug node/<node name>
   $ chroot /host
   ```

   b. Get Device names and make note of the OpenShift Container Storage devices.

   ```
   $ dmsetup ls
   ocs-deviceset-0-data-0-57snx-block-dmcrypt (253:1)
   ```

   c. Remove the mapped device.

   ```
   $ cryptsetup luksClose --debug --verbose ocs-deviceset-0-data-0-57snx-block-dmcrypt
   ```

   If the above command gets stuck due to insufficient privileges, run the following commands:

   - Press **CTRL+Z** to exit the above command.
   - Find PID of the **cryptsetup** process which was stuck.

     ```
     $ ps
     ```

     Example output:

     ```
     PID     TTY    TIME     CMD
     778825   ?     00:00:00 cryptsetup
     ```

     Take a note of the **PID** number to kill. In this example, **PID** is **778825**.

   - Terminate the process using **kill** command.

     ```
     $ kill -9 <PID>
     ```

   - Verify that the device name is removed.

```
$ dmsetup ls
```

7. Delete the namespace and wait till the deletion is complete. You will need to switch to another project if **openshift-storage** is the active project.
   For example:

   ```
   $ oc project default
   $ oc delete project openshift-storage --wait=true --timeout=5m
   ```

   The project is deleted if the following command returns a **NotFound** error.

   ```
   $ oc get project openshift-storage
   ```

   NOTE

   While uninstalling OpenShift Container Storage, if namespace is not deleted completely and remains in **Terminating** state, perform the steps in Troubleshooting and deleting remaining resources during Uninstall to identify objects that are blocking the namespace from being terminated.

8. Delete the local storage operator configurations if you have deployed OpenShift Container Storage using local storage devices. See Removing local storage operator configurations .

9. Unlabel the storage nodes.

   ```
   $ oc label nodes  --all cluster.ocs.openshift.io/openshift-storage-
   $ oc label nodes  --all topology.rook.io/rack-
   ```

10. Remove the OpenShift Container Storage taint if the nodes were tainted.

    ```
    $ oc adm taint nodes --all node.ocs.openshift.io/storage-
    ```

11. Confirm all PVs provisioned using OpenShift Container Storage are deleted. If there is any PV left in the **Released** state, delete it.

    ```
    $ oc get pv
    $ oc delete pv <pv name>
    ```

12. Delete the Multicloud Object Gateway storageclass.

    ```
    $ oc delete storageclass openshift-storage.noobaa.io --wait=true --timeout=5m
    ```

13. Remove **CustomResourceDefinitions**.

    ```
    $ oc delete crd backingstores.noobaa.io bucketclasses.noobaa.io
    cephblockpools.ceph.rook.io cephclusters.ceph.rook.io cephfilesystems.ceph.rook.io
    cephnfses.ceph.rook.io cephobjectstores.ceph.rook.io cephobjectstoreusers.ceph.rook.io
    noobaas.noobaa.io ocsinitializations.ocs.openshift.io storageclusters.ocs.openshift.io
    cephclients.ceph.rook.io cephobjectrealms.ceph.rook.io cephobjectzonegroups.ceph.rook.io
    cephobjectzones.ceph.rook.io cephrbdmirrors.ceph.rook.io --wait=true --timeout=5m
    ```

14. To ensure that OpenShift Container Storage is uninstalled completely, on the OpenShift Container Platform Web Console,

    a. Click **Home → Overview** to access the dashboard.

    b. Verify that the Persistent Storage and Object Service tabs no longer appear next to the **Cluster** tab.

3.1.1. Removing local storage operator configurations

Use the instructions in this section only if you have deployed OpenShift Container Storage using local storage devices.

> **NOTE**
>
> For OpenShift Container Storage deployments only using **localvolume** resources, go directly to step 8.

Procedure

1. Identify the **LocalVolumeSet** and the corresponding **StorageClassName** being used by OpenShift Container Storage.

2. Set the variable SC to the **StorageClass** providing the **LocalVolumeSet**.

```
$ export SC="<StorageClassName>"
```

3. Delete the **LocalVolumeSet**.

```
$ oc delete localvolumesets.local.storage.openshift.io <name-of-volumeset> -n openshift-local-storage
```

4. Delete the local storage PVs for the given **StorageClassName**.

```
$ oc get pv | grep $SC | awk '{print $1}'| xargs oc delete pv
```

5. Delete the **StorageClassName**.

```
$ oc delete sc $SC
```

6. Delete the symlinks created by the **LocalVolumeSet**.

```
[[ ! -z $SC ]] && for i in $(oc get node -l cluster.ocs.openshift.io/openshift-storage= -o jsonpath='{ .items[*].metadata.name }'); do oc debug node/${i} -- chroot /host rm -rfv /mnt/local-storage/${SC}/; done
```

7. Delete **LocalVolumeDiscovery**.

```
$ oc delete localvolumediscovery.local.storage.openshift.io/auto-discover-devices -n openshift-local-storage
```

8. Removing **LocalVolume** resources (if any).

Use the following steps to remove the **LocalVolume** resources that were used to provision PVs in the current or previous OpenShift Container Storage version. Also, ensure that these resources are not being used by other tenants on the cluster.

For each of the local volumes, do the following:

a. Identify the **LocalVolume** and the corresponding **StorageClassName** being used by OpenShift Container Storage.

b. Set the variable LV to the name of the LocalVolume and variable SC to the name of the StorageClass
For example:

```
$ LV=local-block
$ SC=localblock
```

c. Delete the local volume resource.

```
$ oc delete localvolume -n local-storage --wait=true $LV
```

d. Delete the remaining PVs and StorageClasses if they exist.

```
$ oc delete pv -l storage.openshift.com/local-volume-owner-name=${LV} --wait --
timeout=5m
$ oc delete storageclass $SC --wait --timeout=5m
```

e. Clean up the artifacts from the storage nodes for that resource.

```
$ [[ ! -z $SC ]] && for i in $(oc get node -l cluster.ocs.openshift.io/openshift-storage= -o
jsonpath='{ .items[*].metadata.name }'); do oc debug node/${i} -- chroot /host rm -rfv
/mnt/local-storage/${SC}/; done
```

Example output:

```
Starting pod/node-xxx-debug ...
To use host binaries, run `chroot /host`
removed '/mnt/local-storage/localblock/nvme2n1'
removed directory '/mnt/local-storage/localblock'

Removing debug pod ...
Starting pod/node-yyy-debug ...
To use host binaries, run `chroot /host`
removed '/mnt/local-storage/localblock/nvme2n1'
removed directory '/mnt/local-storage/localblock'

Removing debug pod ...
Starting pod/node-zzz-debug ...
To use host binaries, run `chroot /host`
removed '/mnt/local-storage/localblock/nvme2n1'
removed directory '/mnt/local-storage/localblock'

Removing debug pod ...
```

3.2. REMOVING MONITORING STACK FROM OPENSHIFT CONTAINER STORAGE

Use this section to clean up the monitoring stack from OpenShift Container Storage.

The PVCs that are created as a part of configuring the monitoring stack are in the **openshift-monitoring** namespace.

Prerequisites

- PVCs are configured to use OpenShift Container Platform monitoring stack.
  For information, see configuring monitoring stack.

Procedure

1. List the pods and PVCs that are currently running in the **openshift-monitoring** namespace.

```
$ oc get pod,pvc -n openshift-monitoring
NAME                          READY   STATUS    RESTARTS   AGE
pod/alertmanager-main-0        3/3     Running   0          8d
pod/alertmanager-main-1        3/3     Running   0          8d
pod/alertmanager-main-2        3/3     Running   0          8d
pod/cluster-monitoring-
operator-84457656d-pkrxm       1/1     Running   0          8d
pod/grafana-79ccf6689f-2ll28    2/2     Running   0          8d
pod/kube-state-metrics-
7d86fb966-rvd9w                3/3     Running   0          8d
pod/node-exporter-25894        2/2     Running   0          8d
pod/node-exporter-4dsd7        2/2     Running   0          8d
pod/node-exporter-6p4zc        2/2     Running   0          8d
pod/node-exporter-jbjvg        2/2     Running   0          8d
pod/node-exporter-jj4t5        2/2     Running   0          6d18h
pod/node-exporter-k856s        2/2     Running   0          6d18h
pod/node-exporter-rf8gn        2/2     Running   0          8d
pod/node-exporter-rmb5m        2/2     Running   0          6d18h
pod/node-exporter-zj7kx        2/2     Running   0          8d
pod/openshift-state-metrics-
59dbd4f654-4clng               3/3     Running   0          8d
pod/prometheus-adapter-
5df5865596-k8dzn               1/1     Running   0          7d23h
pod/prometheus-adapter-
5df5865596-n2gj9               1/1     Running   0          7d23h
pod/prometheus-k8s-0           6/6     Running   1          8d
pod/prometheus-k8s-1           6/6     Running   1          8d
pod/prometheus-operator-
55cfb858c9-c4zd9               1/1     Running   0          6d21h
pod/telemeter-client-
78fc8fc97d-2rgfp               3/3     Running   0          8d

NAME                                                        STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS                  AGE
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-0   Bound    pvc-0d519c4f-
15a5-11ea-baa0-026d231574aa   40Gi       RWO            ocs-storagecluster-ceph-rbd   8d
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-1   Bound    pvc-
0d5a9825-15a5-11ea-baa0-026d231574aa   40Gi       RWO            ocs-storagecluster-ceph-
```

```
rbd   8d
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-2   Bound    pvc-0d6413dc-15a5-11ea-baa0-026d231574aa   40Gi       RWO            ocs-storagecluster-ceph-rbd   8d
persistentvolumeclaim/my-prometheus-claim-prometheus-k8s-0        Bound    pvc-0b7c19b0-15a5-11ea-baa0-026d231574aa   40Gi       RWO            ocs-storagecluster-ceph-rbd   8d
persistentvolumeclaim/my-prometheus-claim-prometheus-k8s-1        Bound    pvc-0b8aed3f-15a5-11ea-baa0-026d231574aa   40Gi       RWO            ocs-storagecluster-ceph-rbd   8d
```

2. Edit the monitoring **configmap**.

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

3. Remove any **config** sections that reference the OpenShift Container Storage storage classes as shown in the following example and save it.
   Before editing

```
.
.
.
apiVersion: v1
data:
  config.yaml: |
    alertmanagerMain:
      volumeClaimTemplate:
        metadata:
          name: my-alertmanager-claim
        spec:
          resources:
            requests:
              storage: 40Gi
          storageClassName: ocs-storagecluster-ceph-rbd
    prometheusK8s:
      volumeClaimTemplate:
        metadata:
          name: my-prometheus-claim
        spec:
          resources:
            requests:
              storage: 40Gi
          storageClassName: ocs-storagecluster-ceph-rbd
kind: ConfigMap
metadata:
  creationTimestamp: "2019-12-02T07:47:29Z"
  name: cluster-monitoring-config
  namespace: openshift-monitoring
  resourceVersion: "22110"
  selfLink: /api/v1/namespaces/openshift-monitoring/configmaps/cluster-monitoring-config
  uid: fd6d988b-14d7-11ea-84ff-066035b9efa8
.
.
.
```

After editing

```
.
.
.
apiVersion: v1
data:
  config.yaml: |
kind: ConfigMap
metadata:
  creationTimestamp: "2019-11-21T13:07:05Z"
  name: cluster-monitoring-config
  namespace: openshift-monitoring
  resourceVersion: "404352"
  selfLink: /api/v1/namespaces/openshift-monitoring/configmaps/cluster-monitoring-config
  uid: d12c796a-0c5f-11ea-9832-063cd735b81c
.
.
.
```

In this example, **alertmanagerMain** and **prometheusK8s** monitoring components are using the OpenShift Container Storage PVCs.

4. Delete relevant PVCs. Make sure you delete all the PVCs that are consuming the storage classes.

```
$ oc delete -n openshift-monitoring pvc <pvc-name> --wait=true --timeout=5m
```

3.3. REMOVING OPENSHIFT CONTAINER PLATFORM REGISTRY FROM OPENSHIFT CONTAINER STORAGE

Use this section to clean up OpenShift Container Platform registry from OpenShift Container Storage. If you want to configure an alternative storage, see image registry

The PVCs that are created as a part of configuring OpenShift Container Platform registry are in the **openshift-image-registry** namespace.

Prerequisites

- The image registry should have been configured to use an OpenShift Container Storage PVC.

Procedure

1. Edit the **configs.imageregistry.operator.openshift.io** object and remove the content in the **storage** section.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

Before editing

```
.
.
.
storage:
  pvc:
    claim: registry-cephfs-rwx-pvc
.
.
.
```

After editing

```
.
.
.
storage:
  emptyDir: {}
.
.
.
```

In this example, the PVC is called **registry-cephfs-rwx-pvc**, which is now safe to delete.

2. Delete the PVC.

```
$ oc delete pvc <pvc-name> -n openshift-image-registry --wait=true --timeout=5m
```

3.4. REMOVING THE CLUSTER LOGGING OPERATOR FROM OPENSHIFT CONTAINER STORAGE

Use this section to clean up the cluster logging operator from OpenShift Container Storage.

The PVCs that are created as a part of configuring cluster logging operator are in the **openshift-logging** namespace.

Prerequisites

- The cluster logging instance should have been configured to use OpenShift Container Storage PVCs.

Procedure

1. Remove the **ClusterLogging** instance in the namespace.

```
$ oc delete clusterlogging instance -n openshift-logging --wait=true --timeout=5m
```

The PVCs in the **openshift-logging** namespace are now safe to delete.

2. Delete PVCs.

```
$ oc delete pvc <pvc-name> -n openshift-logging --wait=true --timeout=5m
```

CHAPTER 4. STORAGE CLASSES AND STORAGE POOLS

The OpenShift Container Storage operator installs a default storage class depending on the platform in use. This default storage class is owned and controlled by the operator and it cannot be deleted or modified. However, you can create a custom storage class if you want the storage class to have a different behavior.

You can create multiple storage pools which map to storage classes that provide the following features:

- Enable applications with their own high availability to use persistent volumes with two replicas, potentially improving application performance.
- Save space for persistent volume claims using storage classes with compression enabled.

> **NOTE**
>
> Multiple storage classes and multiple pools are not supported for *external mode* OpenShift Container Storage clusters.

> **NOTE**
>
> With a minimal cluster of a single device set, only two new storage classes can be created. Every storage cluster expansion allows two new additional storage classes.

4.1. CREATING STORAGE CLASSES AND POOLS

You can create a storage class using an existing pool or you can create a new pool for the storage class while creating it.

Prerequisites

Ensure that the OpenShift Container Storage cluster is in **Ready** state.

Procedure

1. Log in to OpenShift Web Console.
2. Click **Storage → Storage Classes**.
3. Click **Create Storage Class**
4. Enter the storage class **Name** and **Description**.
5. Select either **Delete** or **Retain** for the Reclaim Policy. By default, **Delete** is selected.
6. Select RBD Provisioner which is the plugin used for provisioning the persistent volumes.
7. You can either create a new pool or use an existing one.

   Create a new pool

   a. Enter a name for the pool.
   b. Choose **2-way-Replication** or **3-way-Replication** as the Data Protection Policy.
   c. Select **Enable compression** if you need to compress the data.

Enabling compression can impact application performance and might prove ineffective when data to be written is already compressed or encrypted. Data written before enabling compression will not be compressed.

d. Click **Create** to create the storage pool.

e. Click **Finish** after the pool is created.

f. Click **Create** to create the storage class.

Use an existing pool

a. Choose a pool from the list.

b. Click **Create** to create the storage class with the selected pool.

CHAPTER 5. CONFIGURE STORAGE FOR OPENSHIFT CONTAINER PLATFORM SERVICES

You can use OpenShift Container Storage to provide storage for OpenShift Container Platform services such as image registry, monitoring, and logging.

The process for configuring storage for these services depends on the infrastructure used in your OpenShift Container Storage deployment.

> **WARNING**
>
> Always ensure that you have plenty of storage capacity for these services. If the storage for these critical services runs out of space, the cluster becomes inoperable and very difficult to recover.
>
> Red Hat recommends configuring shorter curation and retention intervals for these services. See Configuring the Curator schedule and the *Modifying retention time for Prometheus metrics data* sub section of Configuring persistent storage in the OpenShift Container Platform documentation for details.
>
> If you do run out of storage space for these services, contact Red Hat Customer Support.

5.1. CONFIGURING IMAGE REGISTRY TO USE OPENSHIFT CONTAINER STORAGE

OpenShift Container Platform provides a built in Container Image Registry which runs as a standard workload on the cluster. A registry is typically used as a publication target for images built on the cluster as well as a source of images for workloads running on the cluster.

> **WARNING**
>
> This process does not migrate data from an existing image registry to the new image registry. If you already have container images in your existing registry, back up your registry before you complete this process, and re-register your images when this process is complete.

Prerequisites

- You have administrative access to OpenShift Web Console.
- OpenShift Container Storage Operator is installed and running in the **openshift-storage** namespace. In OpenShift Web Console, click **Operators → Installed Operators** to view installed operators.

- Image Registry Operator is installed and running in the **openshift-image-registry** namespace. In OpenShift Web Console, click **Administration → Cluster Settings → Cluster Operators** to view cluster operators.
- A storage class with provisioner **openshift-storage.cephfs.csi.ceph.com** is available. In OpenShift Web Console, click **Storage → Storage Classes**to view available storage classes.

Procedure

1. **Create a Persistent Volume Claim for the Image Registry to use.**
   a. In the OpenShift Web Console, click **Storage → Persistent Volume Claims**.
   b. Set the **Project** to **openshift-image-registry**.
   c. Click **Create Persistent Volume Claim**
      i. From the list of available storage classes retrieved above, specify the **Storage Class** with the provisioner **openshift-storage.cephfs.csi.ceph.com**.
      ii. Specify the Persistent Volume Claim **Name**, for example, **ocs4registry**.
      iii. Specify an **Access Mode** of **Shared Access (RWX)**.
      iv. Specify a **Size** of at least 100 GB.
      v. Click **Create**.
      Wait until the status of the new Persistent Volume Claim is listed as **Bound**.
2. **Configure the cluster's Image Registry to use the new Persistent Volume Claim.**
   a. Click **Administration →Custom Resource Definitions**.
   b. Click the **Config** custom resource definition associated with the **imageregistry.operator.openshift.io** group.
   c. Click the **Instances** tab.
   d. Beside the cluster instance, click the **Action Menu (⋮) → Edit Config**.
   e. Add the new Persistent Volume Claim as persistent storage for the Image Registry.
      i. Add the following under **spec:**, replacing the existing **storage:** section if necessary.

```
storage:
  pvc:
    claim: <new-pvc-name>
```

      For example:

```
storage:
  pvc:
    claim: ocs4registry
```

      ii. Click **Save**.
3. **Verify that the new configuration is being used.**

a. Click **Workloads** → **Pods**.

b. Set the **Project** to **openshift-image-registry**.

c. Verify that the new **image-registry-*** pod appears with a status of **Running**, and that the previous **image-registry-*** pod terminates.

d. Click the new **image-registry-*** pod to view pod details.

e. Scroll down to **Volumes** and verify that the **registry-storage** volume has a **Type** that matches your new Persistent Volume Claim, for example, **ocs4registry**.

5.2. CONFIGURING MONITORING TO USE OPENSHIFT CONTAINER STORAGE

OpenShift Container Storage provides a monitoring stack that is comprised of Prometheus and AlertManager.

Follow the instructions in this section to configure OpenShift Container Storage as storage for the monitoring stack.

IMPORTANT

Monitoring will not function if it runs out of storage space. Always ensure that you have plenty of storage capacity for monitoring.

Red Hat recommends configuring a short retention interval for this service. See the Modifying retention time for Prometheus metrics data of Monitoring guide in the OpenShift Container Platform documentation for details.

Prerequisites

- You have administrative access to OpenShift Web Console.
- OpenShift Container Storage Operator is installed and running in the **openshift-storage** namespace. In OpenShift Web Console, click **Operators** → **Installed Operators** to view installed operators.
- Monitoring Operator is installed and running in the **openshift-monitoring** namespace. In OpenShift Web Console, click **Administration** → **Cluster Settings** → **Cluster Operators** to view cluster operators.
- A storage class with provisioner **openshift-storage.rbd.csi.ceph.com** is available. In OpenShift Web Console, click **Storage** → **Storage Classes**to view available storage classes.

Procedure

1. In the OpenShift Web Console, go to **Workloads** → **Config Maps**.
2. Set the **Project** dropdown to **openshift-monitoring**.
3. Click **Create Config Map**.
4. Define a new **cluster-monitoring-config** Config Map using the following example.

Replace the content in angle brackets (**<**, **>**) with your own values, for example, **retention: 24h** or **storage: 40Gi**.

Replace the **storageClassName** with the **storageclass** that uses the provisioner **openshift-storage.rbd.csi.ceph.com**. In the example given below the name of the **storageclass** is **ocs-storagecluster-ceph-rbd**.

Example **cluster-monitoring-config** Config Map

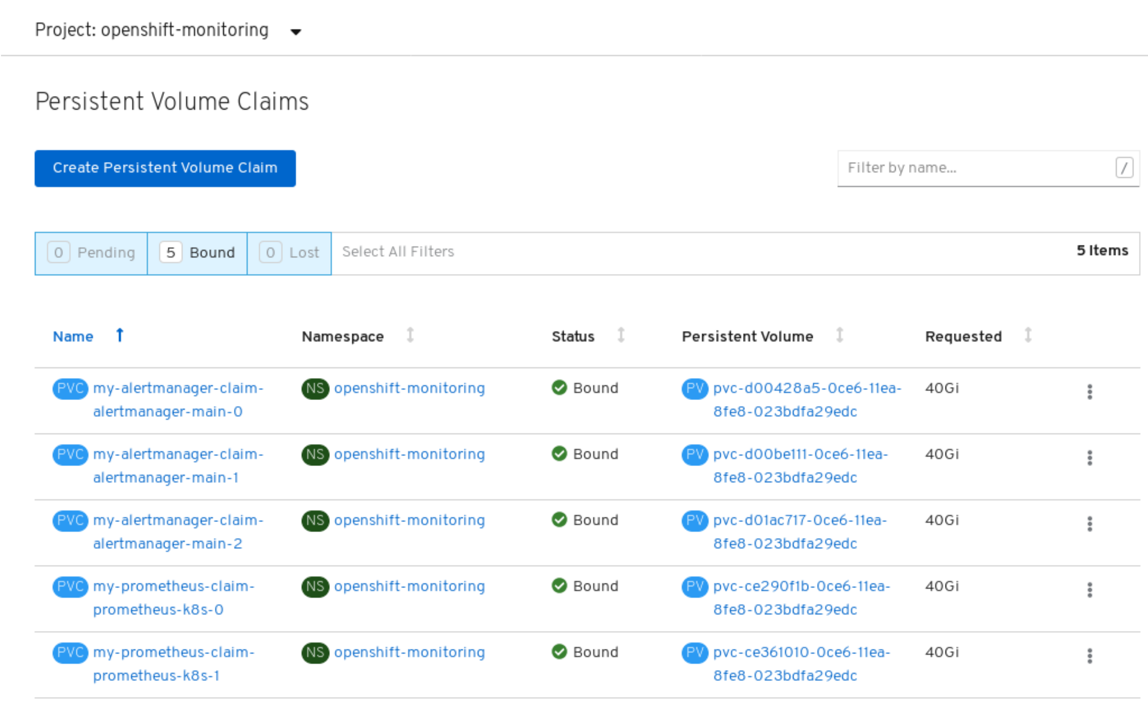
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
      prometheusK8s:
        retention: <time to retain monitoring files, e.g. 24h>
        volumeClaimTemplate:
          metadata:
            name: ocs-prometheus-claim
          spec:
            storageClassName: ocs-storagecluster-ceph-rbd
            resources:
              requests:
                storage: <size of claim, e.g. 40Gi>
      alertmanagerMain:
        volumeClaimTemplate:
          metadata:
            name: ocs-alertmanager-claim
          spec:
            storageClassName: ocs-storagecluster-ceph-rbd
            resources:
              requests:
                storage: <size of claim, e.g. 40Gi>
```

5. Click **Create** to save and create the Config Map.

Verification steps

1. Verify that the Persistent Volume Claims are bound to the pods.
    a. Go to **Storage → Persistent Volume Claims**.
    b. Set the **Project** dropdown to **openshift-monitoring**.
    c. Verify that 5 Persistent Volume Claims are visible with a state of **Bound**, attached to three **alertmanager-main-*** pods, and two **prometheus-k8s-*** pods.

    #### Monitoring storage created and bound



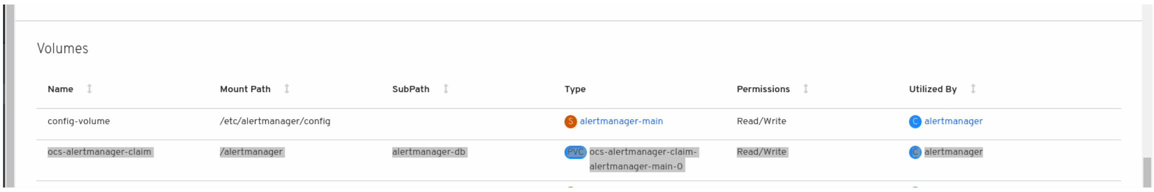
2. Verify that the new **alertmanager-main-*** pods appear with a state of **Running**.

   a. Go to **Workloads → Pods**

   b. Click the new **alertmanager-main-*** pods to view the pod details.

   c. Scroll down to **Volumes** and verify that the volume has a **Type**, **ocs-alertmanager-claim** that matches one of your new Persistent Volume Claims, for example, **ocs-alertmanager-claim-alertmanager-main-0**.

   Persistent Volume Claims attached to alertmanager-main-* pod

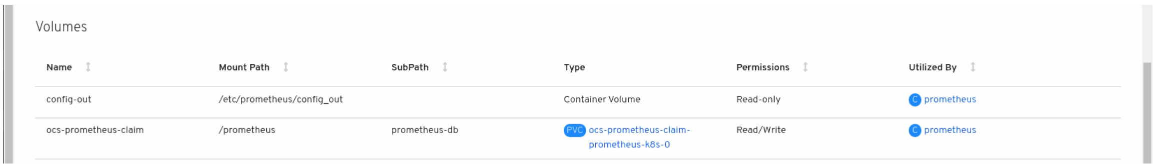


3. Verify that the new **prometheus-k8s-*** pods appear with a state of **Running**.

   a. Click the new **prometheus-k8s-*** pods to view the pod details.

   b. Scroll down to **Volumes** and verify that the volume has a **Type**, **ocs-prometheus-claim** that matches one of your new Persistent Volume Claims, for example, **ocs-prometheus-claim-prometheus-k8s-0**.

   Persistent Volume Claims attached to prometheus-k8s-* pod



5.3. CLUSTER LOGGING FOR OPENSHIFT CONTAINER STORAGE

You can deploy cluster logging to aggregate logs for a range of OpenShift Container Platform services. For information about how to deploy cluster logging, see Deploying cluster logging.

Upon initial OpenShift Container Platform deployment, OpenShift Container Storage is not configured by default and the OpenShift Container Platform cluster will solely rely on default storage available from the nodes. You can edit the default configuration of OpenShift logging (ElasticSearch) to be backed by OpenShift Container Storage to have OpenShift Container Storage backed logging (Elasticsearch).

IMPORTANT

Always ensure that you have plenty of storage capacity for these services. If you run out of storage space for these critical services, the logging application becomes inoperable and very difficult to recover.

Red Hat recommends configuring shorter curation and retention intervals for these services. See Cluster logging curator in the OpenShift Container Platform documentation for details.

If you run out of storage space for these services, contact Red Hat Customer Support.

5.3.1. Configuring persistent storage

You can configure a persistent storage class and size for the Elasticsearch cluster using the storage class name and size parameters. The Cluster Logging Operator creates a Persistent Volume Claim for each data node in the Elasticsearch cluster based on these parameters. For example:

```
spec:
    logStore:
      type: "elasticsearch"
      elasticsearch:
        nodeCount: 3
        storage:
          storageClassName: "ocs-storagecluster-ceph-rbd"
          size: "200G"
```

This example specifies that each data node in the cluster will be bound to a Persistent Volume Claim that requests **200GiB** of **ocs-storagecluster-ceph-rbd** storage. Each primary shard will be backed by a single replica. A copy of the shard is replicated across all the nodes and are always available and the copy can be recovered if at least two nodes exist due to the single redundancy policy. For information about Elasticsearch replication policies, see *Elasticsearch replication policy* in About deploying and configuring cluster logging.

NOTE

Omission of the storage block will result in a deployment backed by default storage. For example:

```
spec:
    logStore:
      type: "elasticsearch"
      elasticsearch:
        nodeCount: 3
        storage: {}
```

For more information, see Configuring cluster logging.

5.3.2. Configuring cluster logging to use OpenShift Container Storage

Follow the instructions in this section to configure OpenShift Container Storage as storage for the OpenShift cluster logging.

> **NOTE**
>
> You can obtain all the logs when you configure logging for the first time in OpenShift Container Storage. However, after you uninstall and reinstall logging, the old logs are removed and only the new logs are processed.

Prerequisites

- You have administrative access to OpenShift Web Console.
- OpenShift Container Storage Operator is installed and running in the **openshift-storage** namespace.
- Cluster logging Operator is installed and running in the **openshift-logging** namespace.

Procedure

1. Click **Administration → Custom Resource Definitions**from the left pane of the OpenShift Web Console.
2. On the Custom Resource Definitions page, click **ClusterLogging**.
3. On the Custom Resource Definition Overview page, select **View Instances** from the Actions menu or click the **Instances** Tab.
4. On the Cluster Logging page, click **Create Cluster Logging**.
   You might have to refresh the page to load the data.
5. In the YAML, replace the **storageClassName** with the **storageclass** that uses the provisioner **openshift-storage.rbd.csi.ceph.com**. In the example given below the name of the **storageclass** is **ocs-storagecluster-ceph-rbd**:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: ocs-storagecluster-ceph-rbd
        size: 200G # Change as per your requirement
      redundancyPolicy: "SingleRedundancy"
  visualization:
```

```
      type: "kibana"
      kibana:
        replicas: 1
    curation:
      type: "curator"
      curator:
        schedule: "30 3 * * *"
    collection:
      logs:
        type: "fluentd"
        fluentd: {}
```

If you have tainted the OpenShift Container Storage nodes, you must add toleration to enable scheduling of the daemonset pods for logging.

```
spec:
[...]
  collection:
    logs:
      fluentd:
        tolerations:
        - effect: NoSchedule
          key: node.ocs.openshift.io/storage
          value: 'true'
      type: fluentd
```

6. Click **Save**.

Verification steps

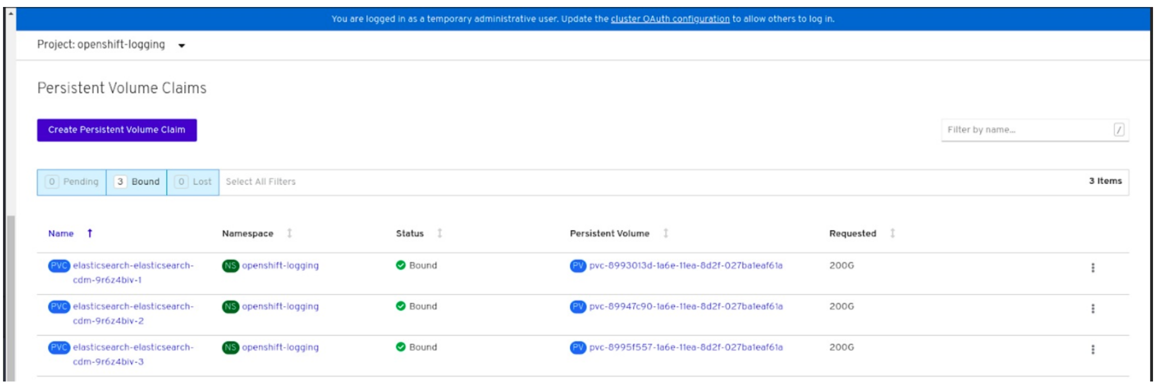
1. Verify that the Persistent Volume Claims are bound to the **elasticsearch** pods.

   a. Go to **Storage → Persistent Volume Claims**

   b. Set the **Project** dropdown to **openshift-logging**.

   c. Verify that Persistent Volume Claims are visible with a state of **Bound**, attached to **elasticsearch-*** pods.

   Figure 5.1. Cluster logging created and bound



2. Verify that the new cluster logging is being used.

   a. Click **Workload → Pods**.

b. Set the Project to **openshift-logging**.

c. Verify that the new **elasticsearch-*** pods appear with a state of **Running**.

d. Click the new **elasticsearch-*** pod to view pod details.

e. Scroll down to **Volumes** and verify that the elasticsearch volume has a **Type** that matches your new Persistent Volume Claim, for example, **elasticsearch-elasticsearch-cdm-9r624biv-3**.

f. Click the Persistent Volume Claim name and verify the storage class name in the PersistenVolumeClaim Overview page.

NOTE

Make sure to use a shorter curator time to avoid PV full scenario on PVs attached to Elasticsearch pods.

You can configure Curator to delete Elasticsearch data based on retention settings. It is recommended that you set the following default index data retention of 5 days as a default.

```
config.yaml: |
    openshift-storage:
      delete:
        days: 5
```

For more details, see Curation of Elasticsearch Data .

NOTE

To uninstall the cluster logging backed by Persistent Volume Claim, use the procedure removing the cluster logging operator from OpenShift Container Storage in the uninstall chapter of the respective deployment guide.

CHAPTER 6. BACKING OPENSHIFT CONTAINER PLATFORM APPLICATIONS WITH OPENSHIFT CONTAINER STORAGE

You cannot directly install OpenShift Container Storage during the OpenShift Container Platform installation. However, you can install OpenShift Container Storage on an existing OpenShift Container Platform by using the Operator Hub and then configure the OpenShift Container Platform applications to be backed by OpenShift Container Storage.

Prerequisites

- OpenShift Container Platform is installed and you have administrative access to OpenShift Web Console.
- OpenShift Container Storage is installed and running in the **openshift-storage** namespace.

Procedure

1. In the OpenShift Web Console, perform one of the following:
    - Click **Workloads → Deployments**.
      In the Deployments page, you can do one of the following:
        - Select any existing deployment and click **Add Storage** option from the **Action** menu (⋮).
        - Create a new deployment and then add storage.
            i. Click **Create Deployment** to create a new deployment.
            ii. Edit the **YAML** based on your requirement to create a deployment.
            iii. Click **Create**.
            iv. Select **Add Storage** from the **Actions** drop down menu on the top right of the page.
    - Click **Workloads → Deployment Configs**
      In the Deployment Configs page, you can do one of the following:
        - Select any existing deployment and click **Add Storage** option from the **Action** menu (⋮).
        - Create a new deployment and then add storage.
            i. Click **Create Deployment Config** to create a new deployment.
            ii. Edit the **YAML** based on your requirement to create a deployment.
            iii. Click **Create**.
            iv. Select **Add Storage** from the **Actions** drop down menu on the top right of the page.
2. In the Add Storage page, you can choose one of the following options:
    - Click the **Use existing claim** option and select a suitable PVC from the drop down list.

- Click the **Create new claim** option.
   a. Select the appropriate **CephFS** or **RBD** storage class from the **Storage Class** drop down list.
   b. Provide a name for the Persistent Volume Claim.
   c. Select ReadWriteOnce (RWO) or ReadWriteMany (RWX) access mode.

      NOTE

      ReadOnlyMany (ROX) is deactivated as it is not supported.

   d. Select the size of the desired storage capacity.

      NOTE

      You can expand block PVs but cannot reduce the storage capacity after the creation of Persistent Volume Claim.

3. Specify the mount path and subpath (if required) for the mount path volume inside the container.
4. Click **Save**.

Verification steps

1. Depending on your configuration, perform one of the following:
   - Click **Workloads → Deployments**.
   - Click **Workloads → Deployment Configs**
2. Set the Project as required.
3. Click the deployment for which you added storage to view the deployment details.
4. Scroll down to **Volumes** and verify that your deployment has a **Type** that matches the Persistent Volume Claim that you assigned.
5. Click the Persistent Volume Claim name and verify the storage class name in the Persistent Volume Claim Overview page.

CHAPTER 7. HOW TO USE DEDICATED WORKER NODES FOR RED HAT OPENSHIFT CONTAINER STORAGE

Using infrastructure nodes to schedule Red Hat OpenShift Container Storage resources saves on Red Hat OpenShift Container Platform subscription costs. Any Red Hat OpenShift Container Platform (RHOCP) node that has an **infra** node-role label requires an OpenShift Container Storage subscription, but not an RHOCP subscription.

It is important to maintain consistency across environments with or without Machine API support. Because of this, it is highly recommended in all cases to have a special category of nodes labeled as either worker or infra or have both roles. See the Section 7.3, “Manual creation of infrastructure nodes” section for more information.

7.1. ANATOMY OF AN INFRASTRUCTURE NODE

Infrastructure nodes for use with OpenShift Container Storage have a few attributes. The **infra** node-role label is required to ensure the node does not consume RHOCP entitlements. The **infra** node-role label is responsible for ensuring only OpenShift Container Storage entitlements are necessary for the nodes running OpenShift Container Storage.

- Labeled with **node-role.kubernetes.io/infra**

Adding an OpenShift Container Storage taint with a **NoSchedule** effect is also required so that the **infra** node will only schedule OpenShift Container Storage resources.

- Tainted with **node.ocs.openshift.io/storage="true"**

The label identifies the RHOCP node as an **infra** node so that RHOCP subscription cost is not applied. The taint prevents non OpenShift Container Storage resources to be scheduled on the tainted nodes.

Example of the taint and labels required on infrastructure node that will be used to run OpenShift Container Storage services:

```
spec:
  taints:
  - effect: NoSchedule
    key: node.ocs.openshift.io/storage
    value: "true"
  metadata:
    creationTimestamp: null
    labels:
      node-role.kubernetes.io/worker: ""
      node-role.kubernetes.io/infra: ""
      cluster.ocs.openshift.io/openshift-storage: ""
```

7.2. MACHINE SETS FOR CREATING INFRASTRUCTURE NODES

If the Machine API is supported in the environment, then labels should be added to the templates for the Machine Sets that will be provisioning the infrastructure nodes. Avoid the anti-pattern of adding labels manually to nodes created by the machine API. Doing so is analogous to adding labels to pods created by a deployment. In both cases, when the pod/node fails, the replacement pod/node will not have the appropriate labels.

> **NOTE**
>
> In EC2 environments, you will need three machine sets, each configured to provision infrastructure nodes in a distinct availability zone (such as us-east-2a, us-east-2b, us-east-2c). Currently, OpenShift Container Storage does not support deploying in more than three availability zones.

The following Machine Set template example creates nodes with the appropriate taint and labels required for infrastructure nodes. This will be used to run OpenShift Container Storage services.

```
template:
  metadata:
    creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: kb-s25vf
      machine.openshift.io/cluster-api-machine-role: worker
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: kb-s25vf-infra-us-west-2a
  spec:
    taints:
    - effect: NoSchedule
      key: node.ocs.openshift.io/storage
      value: "true"
    metadata:
      creationTimestamp: null
      labels:
        node-role.kubernetes.io/infra: ""
        cluster.ocs.openshift.io/openshift-storage: ""
```

7.3. MANUAL CREATION OF INFRASTRUCTURE NODES

Only when the Machine API is not supported in the environment should labels be directly applied to nodes. Manual creation requires that at least 3 RHOCP worker nodes are available to schedule OpenShift Container Storage services, and that these nodes have sufficient CPU and memory resources. To avoid the RHOCP subscription cost, the following is required:

```
oc label node <node> node-role.kubernetes.io/infra=""
oc label node <node> cluster.ocs.openshift.io/openshift-storage=""
```

Adding a **NoSchedule** OpenShift Container Storage taint is also required so that the **infra** node will only schedule OpenShift Container Storage resources and repel any other non-OpenShift Container Storage workloads.

```
oc adm taint node <node> node.ocs.openshift.io/storage="true":NoSchedule
```

WARNING

Do not remove the node-role node-role.kubernetes.io/worker=""

The removal of the **node-role.kubernetes.io/worker=""** can cause issues unless changes are made both to the OpenShift scheduler and to MachineConfig resources.

If already removed, it should be added again to each **infra** node. Adding node-role **node-role.kubernetes.io/infra=""** and OpenShift Container Storage taint is sufficient to conform to entitlement exemption requirements.

CHAPTER 8. SCALING STORAGE NODES

To scale the storage capacity of OpenShift Container Storage, you can do either of the following:

- **Scale up storage nodes** - Add storage capacity to the existing OpenShift Container Storage worker nodes
- **Scale out storage nodes** - Add new worker nodes containing storage capacity

8.1. REQUIREMENTS FOR SCALING STORAGE NODES

Before you proceed to scale the storage nodes, refer to the following sections to understand the node requirements for your specific Red Hat OpenShift Container Storage instance:

- Platform requirements
- Storage device requirements
  - Local storage devices
  - Capacity planning

> **WARNING**
>
> Always ensure that you have plenty of storage capacity.
>
> If storage ever fills completely, it is not possible to add capacity or delete or migrate content away from the storage to free up space. Completely full storage is very difficult to recover.
>
> Capacity alerts are issued when cluster storage capacity reaches 75% (near-full) and 85% (full) of total capacity. Always address capacity warnings promptly, and review your storage regularly to ensure that you do not run out of storage space.
>
> If you do run out of storage space completely, contact Red Hat Customer Support.

8.2. SCALING UP STORAGE BY ADDING CAPACITY TO YOUR OPENSHIFT CONTAINER STORAGE NODES USING LOCAL STORAGE DEVICES

Use this procedure to add storage capacity (additional storage devices) to your configured local storage based OpenShift Container Storage worker nodes on Red Hat Virtualization infrastructures.

Prerequisites

- You must be logged into the OpenShift Container Platform cluster.
- You must have installed the Local Storage Operator. See Install Local Storage Operator

- If you have upgraded from a previous version of OpenShift Container Storage, create a **LocalVolumeSet** object to enable automatic provisioning of devices as described in Post-update configuration changes.

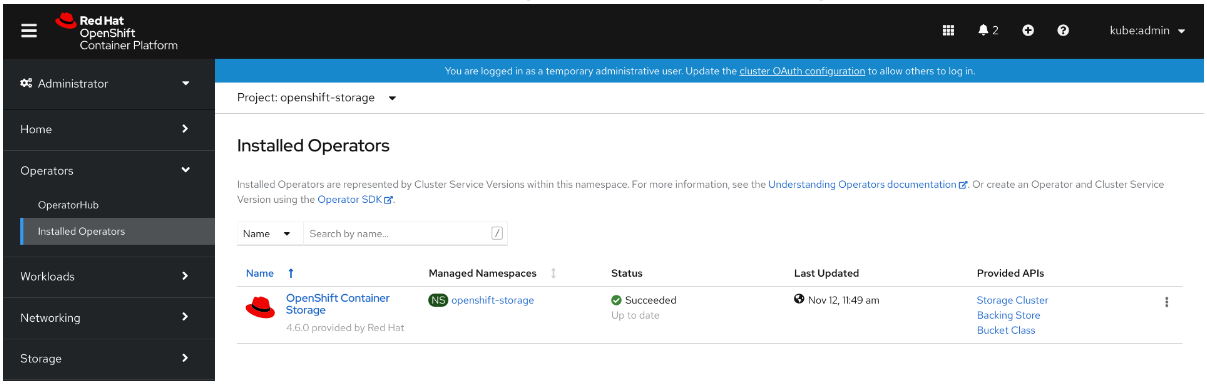
- If you upgraded to OpenShift Container Storage 4.6 from a previous version, ensure that you have followed post-upgrade procedures to create the **LocalVolumeDiscovery** object. See Post-update configuration changes for details.

- You must have three OpenShift Container Platform worker nodes with the same storage type and size attached to each node (for example, 2TB NVMe drive) as the original OpenShift Container Storage StorageCluster was created with.

Procedure

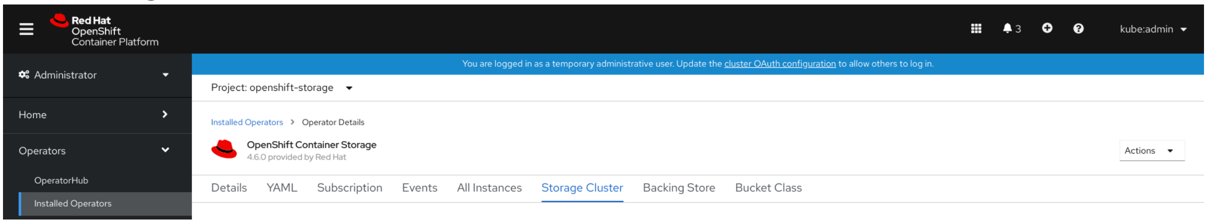
To add capacity, you can either use a storage class that you provisioned during the deployment or any other storage class that matches the filter.

1. On the OpenShift web console, click on **Operators → Installed Operators**.



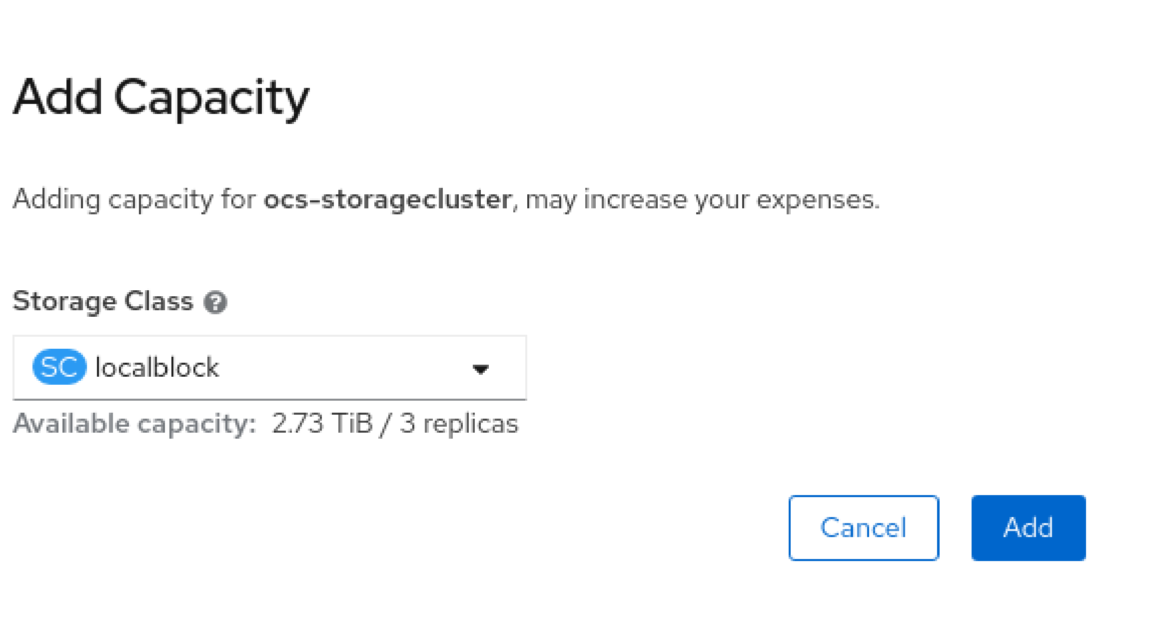
2. Click **OpenShift Container Storage** Operator.

3. Click **Storage Cluster** tab.



4. The visible list should have only one item. Click (⋮) on the far right to extend the options menu.

5. Select **Add Capacity** from the options menu.

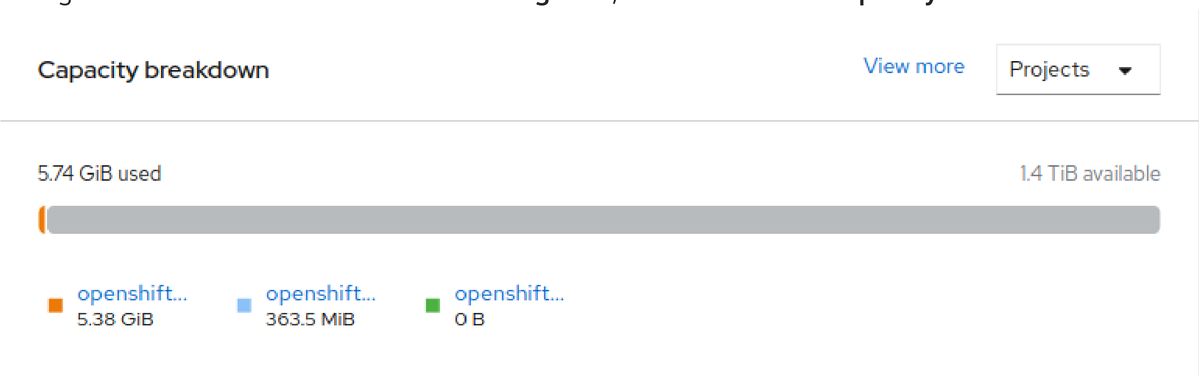


6. Select the **Storage Class** for which you added disks or the new storage class depending on your requirement. Available Capacity displayed is based on the local disks available in storage class.

7. Click **Add**.
   You might need to wait a couple of minutes for the storage cluster to reach **Ready** state.

Verification steps

- Navigate to **Overview → Persistent Storage** tab, then check the **Capacity breakdown** card.



  Note that the capacity increases based on your selections.

- Verify that the new OSDs and their corresponding new PVCs are created.
  - To view the state of the newly created OSDs:
    a. Click **Workloads → Pods** from the OpenShift Web Console.
    b. Select **openshift-storage** from the **Project** drop-down list.
  - To view the state of the PVCs:
    a. Click **Storage → Persistent Volume Claims** from the OpenShift Web Console.
    b. Select **openshift-storage** from the **Project** drop-down list.
- (Optional) If data encryption is enabled on the cluster, verify that the new OSD devices are encrypted.

a. Identify the node(s) where the new OSD pod(s) are running.

```
$ oc get -o=custom-columns=NODE:.spec.nodeName pod/<OSD pod name>
```

For example:

```
oc get -o=custom-columns=NODE:.spec.nodeName pod/rook-ceph-osd-0-544db49d7f-qrgqm
```

b. For each of the nodes identified in previous step, do the following:

i. Create a debug pod and open a chroot environment for the selected host(s).

```
$ oc debug node/<node name>
$ chroot /host
```

ii. Run “lsblk” and check for the “crypt” keyword beside the **ocs-deviceset** name(s)

```
$ lsblk
```

IMPORTANT

OpenShift Container Storage does not support cluster reduction either by reducing OSDs or reducing nodes.

8.3. SCALING OUT STORAGE CAPACITY BY ADDING NEW NODES

To scale out storage capacity, you need to perform the following:

- Add a new node to increase the storage capacity when existing worker nodes are already running at their maximum supported OSDs, which is the increment of 3 OSDs of the capacity selected during initial configuration.
- Verify that the new node is added successfully
- Scale up the storage capacity after the node is added

8.3.1. Adding a node using a local storage device

Use this procedure to add a node on Red Hat Virtualization infrastructures.

Prerequisites

- You must be logged into the OpenShift Container Platform (RHOCP) cluster.
- If you upgraded to OpenShift Container Storage 4.6 from a previous version, ensure that you have followed post-upgrade procedures to create the **LocalVolumeDiscovery** object. See Post-update configuration changes for details.
- You must have three OpenShift Container Platform worker nodes with the same storage type and size attached to each node (for example, 2TB SSD or 2TB NVMe drive) as the original OpenShift Container Storage StorageCluster was created with.

- If you have upgraded from a previous version of OpenShift Container Storage, create a LocalVolumeSet object to enable automatic provisioning of devices as described in Post-update configuration changes.

Procedure

1. Create a new VM on Red Hat Virtualization with the required infrastructure. See Platform requirements.

2. Create a new OpenShift Container Platform worker node using the new VM.

3. Check for certificate signing requests (CSRs) related to OpenShift Container Storage that are in **Pending** state:

```
$ oc get csr
```

4. Approve all required OpenShift Container Storage CSRs for the new node:

```
$ oc adm certificate approve <Certificate_Name>
```

5. Click **Compute → Nodes**, confirm if the new node is in **Ready** state.

6. Apply the OpenShift Container Storage label to the new node using any one of the following:

   From User interface

   a. For the new node, click **Action Menu (⋮) → Edit Labels**

   b. Add **cluster.ocs.openshift.io/openshift-storage** and click **Save**.

   From Command line interface

   - Execute the following command to apply the OpenShift Container Storage label to the new node:

```
$ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
```

7. Click **Operators → Installed Operators** from the OpenShift Web Console.
From the **Project** drop-down list, make sure to select the project where the Local Storage Operator is installed.

8. Click on **Local Storage**.

9. Click the **Local Volume Discovery** tab

10. Beside the **LocalVolumeDiscovery**, click Action menu **(⋮)** → Edit **Local Volume Discovery**.

11. In the YAML, add the hostname of the new node in the values field under the **node selector**.

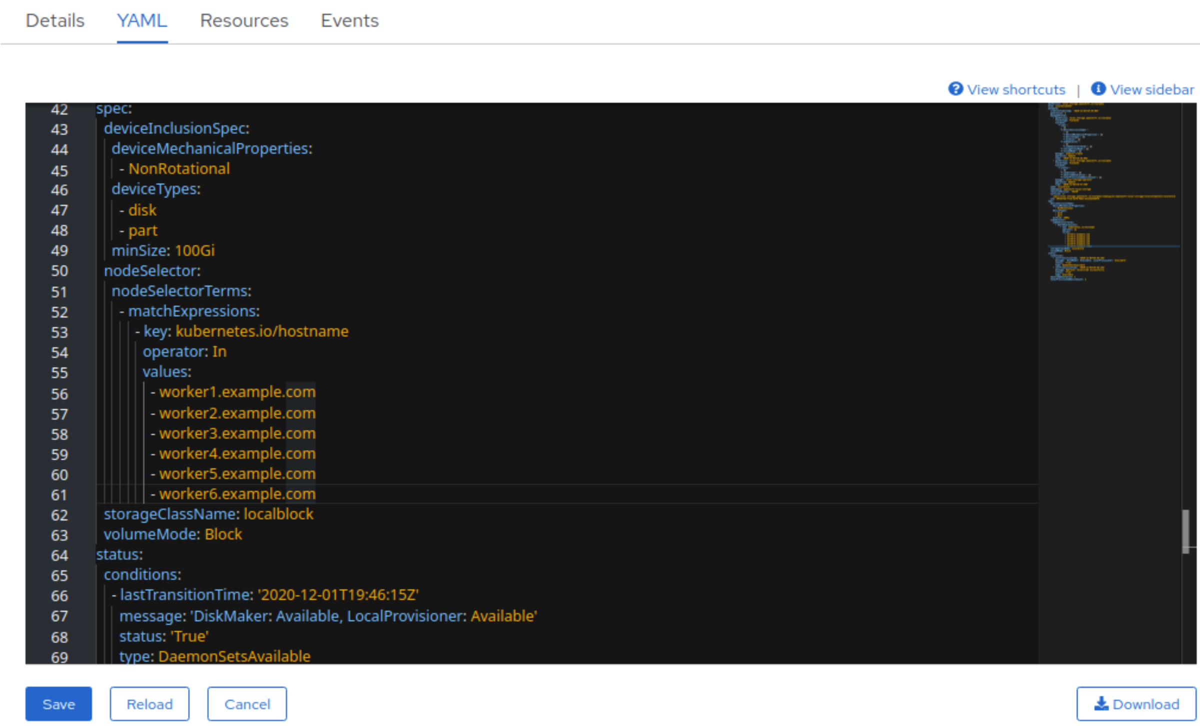
12. Click **Save**.

13. Click the **Local Volume Sets** tab.

14. Beside the **LocalVolumeSet**, click Action menu **(⋮)** → **Edit Local Volume Set**

15. In the YAML, add the hostname of the new node in the **values** field under the **node selector**.

Figure 8.1. YAML showing the addition of new hostnames



16. Click **Save**.

> **NOTE**
>
> It is recommended to add 3 nodes each in different zones. You must add 3 nodes and perform this procedure for all of them.

Verification steps

- To verify that the new node is added, see Verifying the addition of a new node .

8.3.2. Verifying the addition of a new node

1. Execute the following command and verify that the new node is present in the output:

```
$ oc get nodes --show-labels | grep cluster.ocs.openshift.io/openshift-storage= |cut -d' ' -f1
```

2. Click **Workloads → Pods**, confirm that at least the following pods on the new node are in **Running** state:

   - **csi-cephfsplugin-***
   - **csi-rbdplugin-***

8.3.3. Scaling up storage capacity

After you add a new node to OpenShift Container Storage, you must scale up the storage capacity as described in Scaling up storage by adding capacity .

CHAPTER 9. MULTICLOUD OBJECT GATEWAY

9.1. ABOUT THE MULTICLOUD OBJECT GATEWAY

The Multicloud Object Gateway (MCG) is a lightweight object storage service for OpenShift, allowing users to start small and then scale as needed on-premise, in multiple clusters, and with cloud-native storage.

9.2. ACCESSING THE MULTICLOUD OBJECT GATEWAY WITH YOUR APPLICATIONS

You can access the object service with any application targeting AWS S3 or code that uses AWS S3 Software Development Kit (SDK). Applications need to specify the MCG endpoint, an access key, and a secret access key. You can use your terminal or the MCG CLI to retrieve this information.

Prerequisites

- A running OpenShift Container Storage Platform
- Download the MCG command-line interface for easier management:

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

- Alternatively, you can install the **mcg** package from the OpenShift Container Storage RPMs found at Download RedHat OpenShift Container Storage page .

You can access the relevant endpoint, access key, and secret access key two ways:

- Section 9.2.1, “Accessing the Multicloud Object Gateway from the terminal”
- Section 9.2.2, “Accessing the Multicloud Object Gateway from the MCG command-line interface”

Accessing the MCG bucket(s) using the virtual-hosted style

> **Example 9.1. Example**
>
> If the client application tries to access https://<bucket-name>.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
>
> where **<bucket-name>** is the name of the MCG bucket
>
> For example, https://mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
>
> A DNS entry is needed for **mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com** to point to the S3 Service.

IMPORTANT

Ensure that you have a DNS entry in order to point the client application to the MCG bucket(s) using the virtual-hosted style.

9.2.1. Accessing the Multicloud Object Gateway from the terminal

Procedure

Run the **describe** command to view information about the MCG endpoint, including its access key (**AWS_ACCESS_KEY_ID** value) and secret access key (**AWS_SECRET_ACCESS_KEY** value):

```
# oc describe noobaa -n openshift-storage
```

The output will look similar to the following:

```
Name:         noobaa
Namespace:    openshift-storage
Labels:       <none>
Annotations:  <none>
API Version:  noobaa.io/v1alpha1
Kind:         NooBaa
Metadata:
  Creation Timestamp:  2019-07-29T16:22:06Z
  Generation:          1
  Resource Version:    6718822
  Self Link:           /apis/noobaa.io/v1alpha1/namespaces/openshift-storage/noobaas/noobaa
  UID:                 019cfb4a-b21d-11e9-9a02-06c8de012f9e
Spec:
Status:
  Accounts:
    Admin:
      Secret Ref:
        Name:           noobaa-admin
        Namespace:      openshift-storage
  Actual Image:         noobaa/noobaa-core:4.0
  Observed Generation:  1
  Phase:                Ready
  Readme:

  Welcome to NooBaa!
  -----------------

  Welcome to NooBaa!
    -----------------
    NooBaa Core Version:
    NooBaa Operator Version:

    Lets get started:

    1. Connect to Management console:

      Read your mgmt console login information (email & password) from secret: "noobaa-admin".

        kubectl get secret noobaa-admin -n openshift-storage -o json | jq '.data|map_values(@base64d)'
```

```
    Open the management console service - take External IP/DNS or Node Port or use port
forwarding:

      kubectl port-forward -n openshift-storage service/noobaa-mgmt 11443:443 &
      open https://localhost:11443

  2. Test S3 client:

    kubectl port-forward -n openshift-storage service/s3 10443:443 &
(1)
    NOOBAA_ACCESS_KEY=$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r
'.data.AWS_ACCESS_KEY_ID|@base64d')
(2)
    NOOBAA_SECRET_KEY=$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r
'.data.AWS_SECRET_ACCESS_KEY|@base64d')
    alias s3='AWS_ACCESS_KEY_ID=$NOOBAA_ACCESS_KEY
AWS_SECRET_ACCESS_KEY=$NOOBAA_SECRET_KEY aws --endpoint https://localhost:10443 --
no-verify-ssl s3'
    s3 ls


  Services:
    Service Mgmt:
      External DNS:
        https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
        https://a3406079515be11eaa3b70683061451e-1194613580.us-east-
2.elb.amazonaws.com:443
      Internal DNS:
        https://noobaa-mgmt.openshift-storage.svc:443
      Internal IP:
        https://172.30.235.12:443
      Node Ports:
        https://10.0.142.103:31385
      Pod Ports:
        https://10.131.0.19:8443
    serviceS3:
      External DNS: (3)
        https://s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
        https://a340f4e1315be11eaa3b70683061451e-943168195.us-east-2.elb.amazonaws.com:443
      Internal DNS:
        https://s3.openshift-storage.svc:443
      Internal IP:
        https://172.30.86.41:443
      Node Ports:
        https://10.0.142.103:31011
      Pod Ports:
        https://10.131.0.19:6443
```

1. access key (**AWS_ACCESS_KEY_ID** value)
2. secret access key (**AWS_SECRET_ACCESS_KEY** value)
3. MCG endpoint

> **NOTE**
>
> The output from the **oc describe noobaa** command lists the internal and external DNS names that are available. When using the internal DNS, the traffic is free. The external DNS uses Load Balancing to process the traffic, and therefore has a cost per hour.

9.2.2. Accessing the Multicloud Object Gateway from the MCG command-line interface

Prerequisites

- Download the MCG command-line interface:

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

Procedure

Run the **status** command to access the endpoint, access key, and secret access key:

```
noobaa status -n openshift-storage
```

The output will look similar to the following:

```
INFO[0000] Namespace: openshift-storage
INFO[0000]
INFO[0000] CRD Status:
INFO[0003]   Exists: CustomResourceDefinition "noobaas.noobaa.io"
INFO[0003]   Exists: CustomResourceDefinition "backingstores.noobaa.io"
INFO[0003]   Exists: CustomResourceDefinition "bucketclasses.noobaa.io"
INFO[0004]   Exists: CustomResourceDefinition "objectbucketclaims.objectbucket.io"
INFO[0004]   Exists: CustomResourceDefinition "objectbuckets.objectbucket.io"
INFO[0004]
INFO[0004] Operator Status:
INFO[0004]   Exists: Namespace "openshift-storage"
INFO[0004]   Exists: ServiceAccount "noobaa"
INFO[0005]   Exists: Role "ocs-operator.v0.0.271-6g45f"
INFO[0005]   Exists: RoleBinding "ocs-operator.v0.0.271-6g45f-noobaa-f9vpj"
INFO[0006]   Exists: ClusterRole "ocs-operator.v0.0.271-fjhgh"
INFO[0006]   Exists: ClusterRoleBinding "ocs-operator.v0.0.271-fjhgh-noobaa-pdxn5"
INFO[0006]   Exists: Deployment "noobaa-operator"
INFO[0006]
INFO[0006] System Status:
INFO[0007]   Exists: NooBaa "noobaa"
INFO[0007]   Exists: StatefulSet "noobaa-core"
INFO[0007]   Exists: Service "noobaa-mgmt"
INFO[0008]   Exists: Service "s3"
INFO[0008]   Exists: Secret "noobaa-server"
INFO[0008]   Exists: Secret "noobaa-operator"
INFO[0008]   Exists: Secret "noobaa-admin"
INFO[0009]   Exists: StorageClass "openshift-storage.noobaa.io"
INFO[0009]   Exists: BucketClass "noobaa-default-bucket-class"
INFO[0009]   (Optional) Exists: BackingStore "noobaa-default-backing-store"
INFO[0010]   (Optional) Exists: CredentialsRequest "noobaa-cloud-creds"
```

```
INFO[0010]  (Optional) Exists: PrometheusRule "noobaa-prometheus-rules"
INFO[0010]  (Optional) Exists: ServiceMonitor "noobaa-service-monitor"
INFO[0011]  (Optional) Exists: Route "noobaa-mgmt"
INFO[0011]  (Optional) Exists: Route "s3"
INFO[0011]  Exists: PersistentVolumeClaim "db-noobaa-core-0"
INFO[0011]  System Phase is "Ready"
INFO[0011]  Exists:  "noobaa-admin"

#------------------#
#- Mgmt Addresses -#
#------------------#

ExternalDNS : [https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a3406079515be11eaa3b70683061451e-1194613580.us-east-2.elb.amazonaws.com:443]
ExternalIP  : []
NodePorts   : [https://10.0.142.103:31385]
InternalDNS : [https://noobaa-mgmt.openshift-storage.svc:443]
InternalIP  : [https://172.30.235.12:443]
PodPorts    : [https://10.131.0.19:8443]

#--------------------#
#- Mgmt Credentials -#
#--------------------#

email    : admin@noobaa.io
password : HKLbH1rSuVU0I/souIkSiA==

#----------------#
#- S3 Addresses -#
#----------------#

1
ExternalDNS : [https://s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a340f4e1315be11eaa3b70683061451e-943168195.us-east-2.elb.amazonaws.com:443]
ExternalIP  : []
NodePorts   : [https://10.0.142.103:31011]
InternalDNS : [https://s3.openshift-storage.svc:443]
InternalIP  : [https://172.30.86.41:443]
PodPorts    : [https://10.131.0.19:6443]

#------------------#
#- S3 Credentials -#
#------------------#

2
AWS_ACCESS_KEY_ID     : jVmAsu9FsvRHYmfjTiHV
3
AWS_SECRET_ACCESS_KEY : E//420VNedJfATvVSmDz6FMtsSAzuBv6z180PT5c

#------------------#
#- Backing Stores -#
#------------------#

NAME                        TYPE    TARGET-BUCKET                                         PHASE   AGE
noobaa-default-backing-store   aws-s3   noobaa-backing-store-15dc896d-7fe0-4bed-9349-
5942211b93c9   Ready   141h35m32s
```

```
#------------------#
#- Bucket Classes -#
#------------------#

NAME                        PLACEMENT                                                      PHASE   AGE
noobaa-default-bucket-class   {Tiers:[{Placement: BackingStores:[noobaa-default-backing-store]}]}
Ready   141h35m33s

#-----------------#
#- Bucket Claims -#
#-----------------#

No OBC's found.
```

1. endpoint
2. access key
3. secret access key

You now have the relevant endpoint, access key, and secret access key in order to connect to your applications.

Example 9.2. Example

If AWS S3 CLI is the application, the following command will list buckets in OpenShift Container Storage:

```
AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>
AWS_SECRET_ACCESS_KEY=<AWS_SECRET_ACCESS_KEY>
aws --endpoint <ENDPOINT> --no-verify-ssl s3 ls
```

9.3. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD

9.3.1. Adding storage resources for hybrid or Multicloud using the MCG command line interface

The Multicloud Object Gateway (MCG) simplifies the process of spanning data across cloud provider and clusters.

To do so, add a backing storage that can be used by the MCG.

Prerequisites

- Download the MCG command-line interface:

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

- Alternatively, you can install the **mcg** package from the OpenShift Container Storage RPMs found here Download RedHat OpenShift Container Storage page .

Procedure

1. From the MCG command-line interface, run the following command:

```
noobaa backingstore create <backing-store-type> <backingstore_name> --access-key=
<AWS ACCESS KEY> --secret-key=<AWS SECRET ACCESS KEY> --target-bucket
<bucket-name>
```

   a. Replace **<backing-store-type>** with your relevant backing store type: **aws-s3**, **google-cloud-store**, **azure-blob**, **s3-compatible**, or **ibm-cos**.

   b. Replace **<backingstore_name>** with the name of the backingstore.

   c. Replace **<AWS ACCESS KEY>** and **<AWS SECRET ACCESS KEY>** with an AWS access key ID and secret access key you created for this purpose.

   d. Replace **<bucket-name>** with an existing AWS bucket name. This argument tells NooBaa which bucket to use as a target bucket for its backing store, and subsequently, data storage and administration.
   The output will be similar to the following:

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "aws-resource"
INFO[0002]   Created: Secret "backing-store-secret-aws-resource"
```

You can also add storage resources using a YAML:

1. Create a secret with the credentials:

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

   a. You must supply and encode your own AWS access key ID and secret access key using Base64, and use the results in place of **<AWS ACCESS KEY ID ENCODED IN BASE64>** and **<AWS SECRET ACCESS KEY ENCODED IN BASE64>**.

   b. Replace **<backingstore-secret-name>** with a unique name.

2. Apply the following YAML for a specific backing store:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: noobaa
```

```
spec:
  awsS3:
    secret:
      name: <backingstore-secret-name>
      namespace: noobaa
    targetBucket: <bucket-name>
  type: <backing-store-type>
```

a. Replace **<bucket-name>** with an existing AWS bucket name. This argument tells NooBaa which bucket to use as a target bucket for its backing store, and subsequently, data storage and administration.

b. Replace **<backingstore-secret-name>** with the name of the secret created in the previous step.

c. Replace <backing-store-type> with your relevant backing store type: **aws-s3**, **google-cloud-store**, **azure-blob**, **s3-compatible**, or **ibm-cos**.

9.3.2. Creating an s3 compatible Multicloud Object Gateway backingstore

The Multicloud Object Gateway can use any S3 compatible object storage as a backing store, for example, Red Hat Ceph Storage's RADOS Gateway (RGW). The following procedure shows how to create an S3 compatible Multicloud Object Gateway backing store for Red Hat Ceph Storage's RADOS Gateway. Note that when RGW is deployed, Openshift Container Storage operator creates an S3 compatible backingstore for Multicloud Object Gateway automatically.

Procedure

1. From the Multicloud Object Gateway (MCG) command-line interface, run the following NooBaa command:

```
noobaa backingstore create s3-compatible rgw-resource --access-key=<RGW ACCESS KEY> --secret-key=<RGW SECRET KEY> --target-bucket=<bucket-name> --endpoint=<RGW endpoint>
```

a. To get the **<RGW ACCESS KEY>** and **<RGW SECRET KEY>**, run the following command using your RGW user secret name:

```
oc get secret <RGW USER SECRET NAME> -o yaml
```

b. Decode the access key ID and the access key from Base64 and keep them.

c. Replace **<RGW USER ACCESS KEY>** and **<RGW USER SECRET ACCESS KEY>** with the appropriate, decoded data from the previous step.

d. Replace **<bucket-name>** with an existing RGW bucket name. This argument tells Multicloud Object Gateway which bucket to use as a target bucket for its backing store, and subsequently, data storage and administration.

e. To get the **<RGW endpoint>**, see Accessing the RADOS Object Gateway S3 endpoint . The output will be similar to the following:

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "rgw-resource"
INFO[0002]   Created: Secret "backing-store-secret-rgw-resource"
```

–

You can also create the backingstore using a YAML:

1. Create a **CephObjectStore** user. This also creates a secret containing the RGW credentials:

```
apiVersion: ceph.rook.io/v1
kind: CephObjectStoreUser
metadata:
  name: <RGW-Username>
  namespace: openshift-storage
spec:
  store: ocs-storagecluster-cephobjectstore
  displayName: "<Display-name>"
```

   a. Replace **<RGW-Username>** and **<Display-name>** with a unique username and display name.

2. Apply the following YAML for an S3-Compatible backing store:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <backingstore-name>
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <RGW endpoint>
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    signatureVersion: v4
    targetBucket: <RGW-bucket-name>
  type: s3-compatible
```

   a. Replace **<backingstore-secret-name>** with the name of the secret that was created with **CephObjectStore** in the previous step.

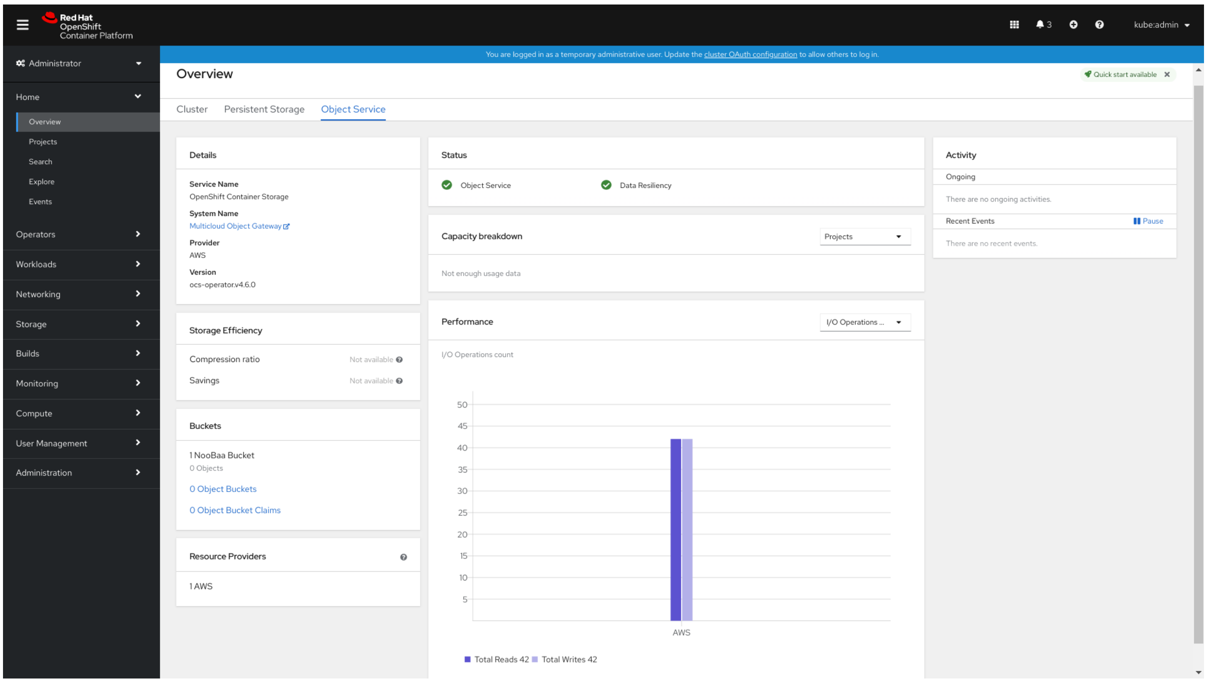
   b. Replace **<bucket-name>** with an existing RGW bucket name. This argument tells Multicloud Object Gateway which bucket to use as a target bucket for its backing store, and subsequently, data storage and administration.

   c. To get the **<RGW endpoint>**, see Accessing the RADOS Object Gateway S3 endpoint .

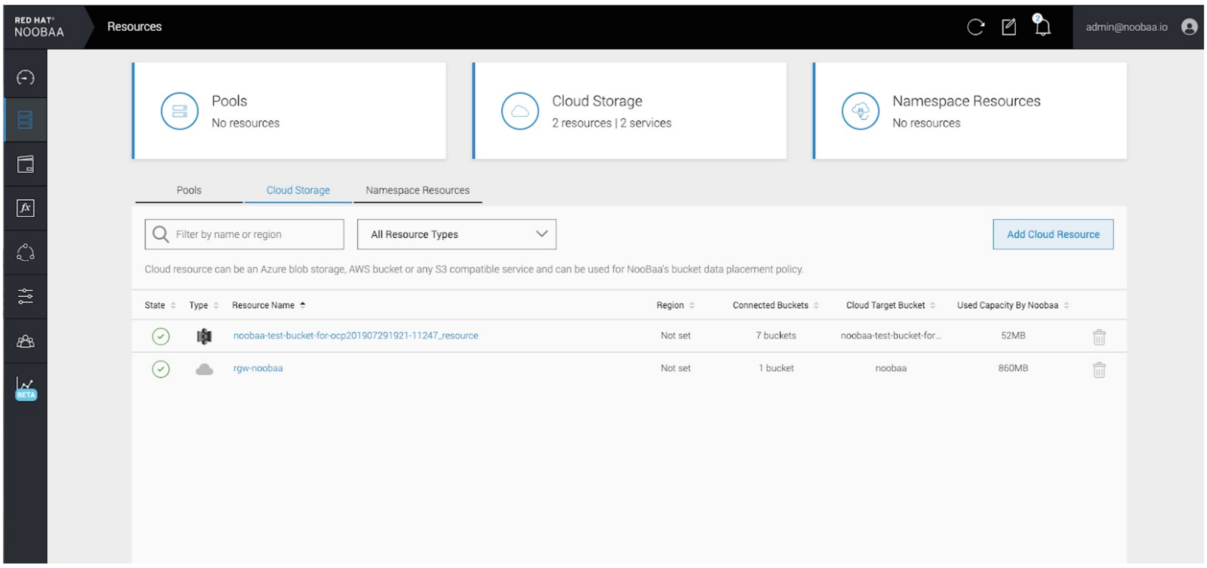
9.3.3. Adding storage resources for hybrid and Multicloud using the user interface

Procedure

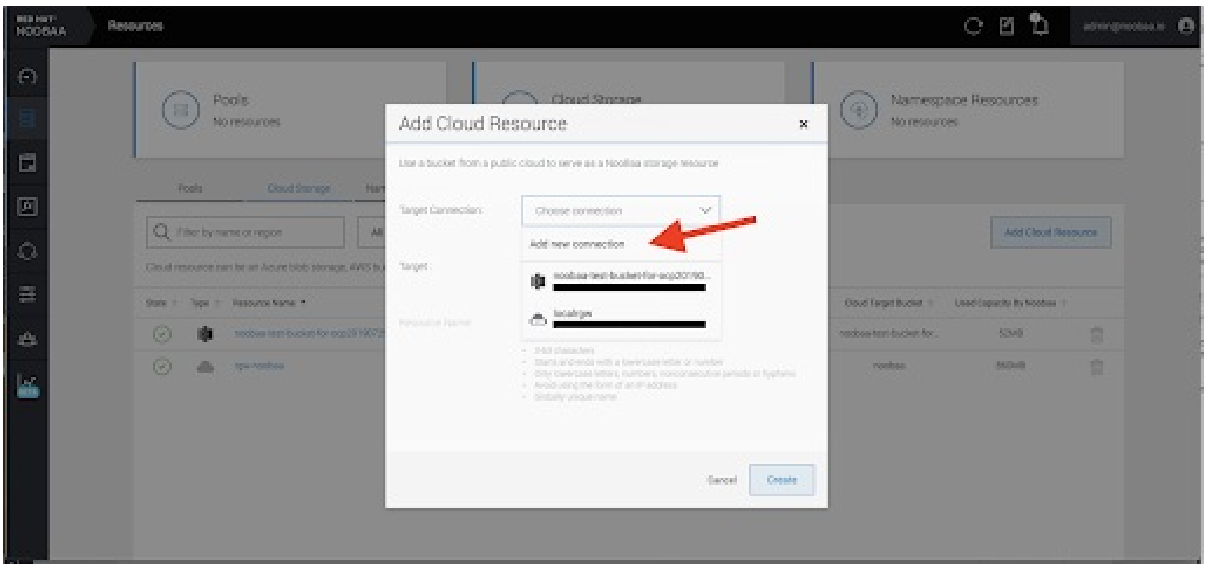
1. In your OpenShift Storage console, navigate to **Overview → Object Service →** select the **Multicloud Object Gateway** link:



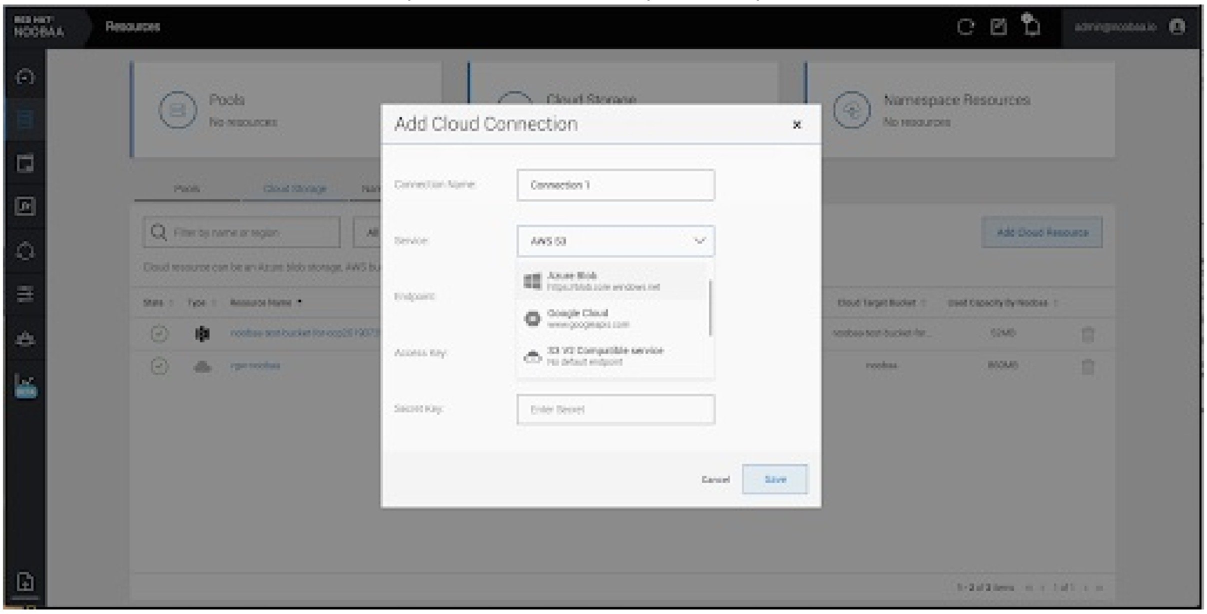
2. Select the **Resources** tab in the left, highlighted below. From the list that populates, select **Add Cloud Resource**:



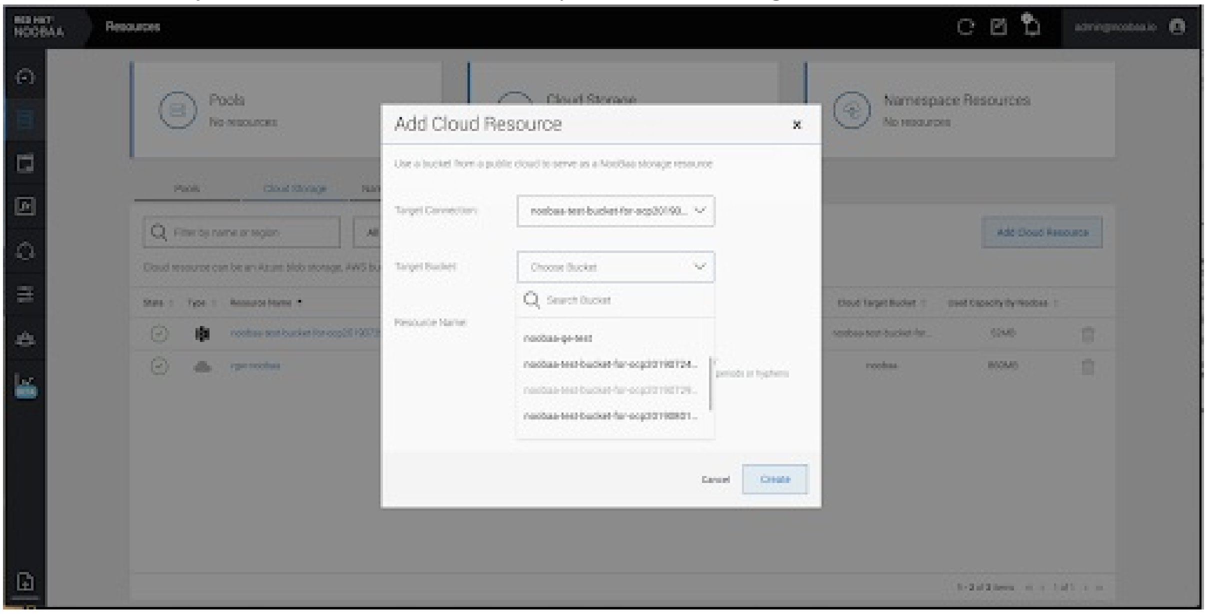
3. Select **Add new connection**:



4. Select the relevant native cloud provider or S3 compatible option and fill in the details:



5. Select the newly created connection and map it to the existing bucket:



6. Repeat these steps to create as many backing stores as needed.

NOTE

Resources created in NooBaa UI cannot be used by OpenShift UI or MCG CLI.

9.3.4. Creating a new bucket class

Bucket class is a CRD representing a class of buckets that defines tiering policies and data placements for an Object Bucket Class (OBC).

Use this procedure to create a bucket class in OpenShift Container Storage.

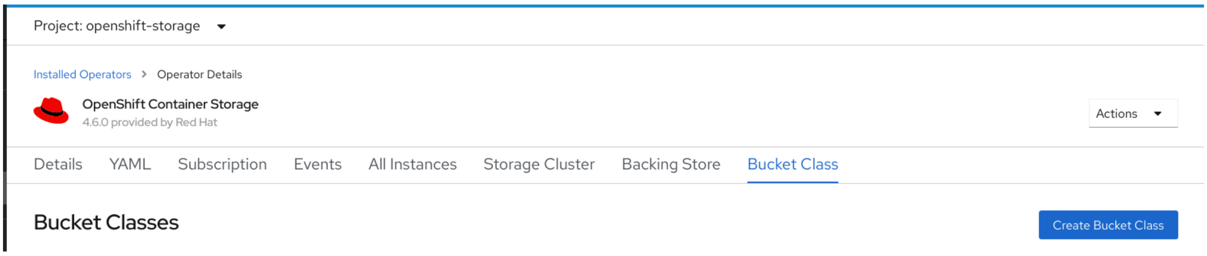
Procedure

1. Click **Operators → Installed Operators** from the left pane of the OpenShift Web Console to view the installed operators.

2. Click **OpenShift Container Storage** Operator.

3. On the OpenShift Container Storage Operator page, scroll right and click the **Bucket Class** tab.

Figure 9.1. OpenShift Container Storage Operator page with Bucket Class tab

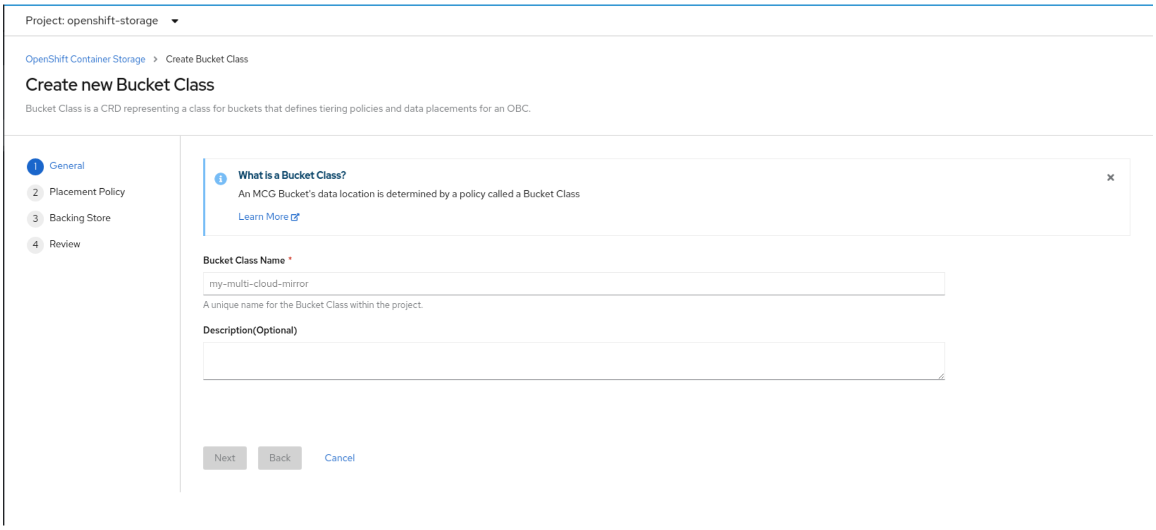


4. Click **Create Bucket Class**.

5. On the Create new Bucket Class page, perform the following:

   a. Enter a **Bucket Class Name** and click **Next**.

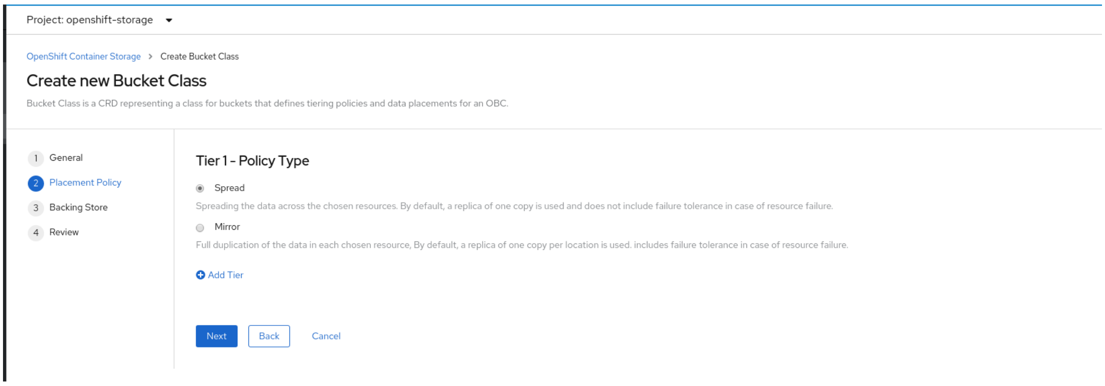
   Figure 9.2. Create Bucket Class page

   b. In Placement Policy, select **Tier 1 - Policy Type** and click **Next**. You can choose either one of the options as per your requirements.

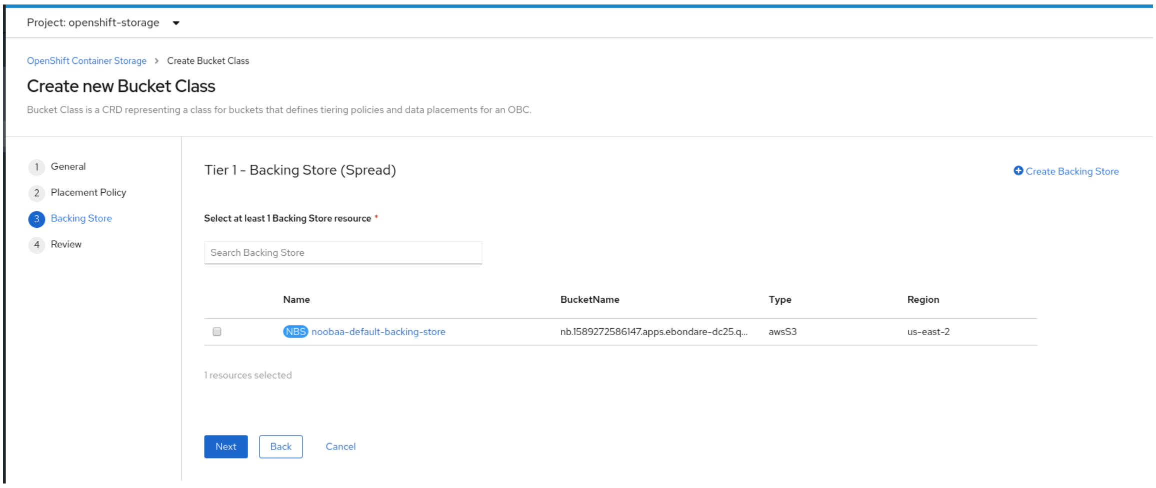
      - **Spread** allows spreading of the data across the chosen resources.
      - **Mirror** allows full duplication of the data across the chosen resources.
      - Click **Add Tier** to add another policy tier.

Figure 9.3. Tier 1 - Policy Type selection page



c. Select atleast one **Backing Store** resource from the available list if you have selected Tier 1 - Policy Type as Spread and click **Next**. Alternatively, you can also create a new backing store.

Figure 9.4. Tier 1 - Backing Store selection page

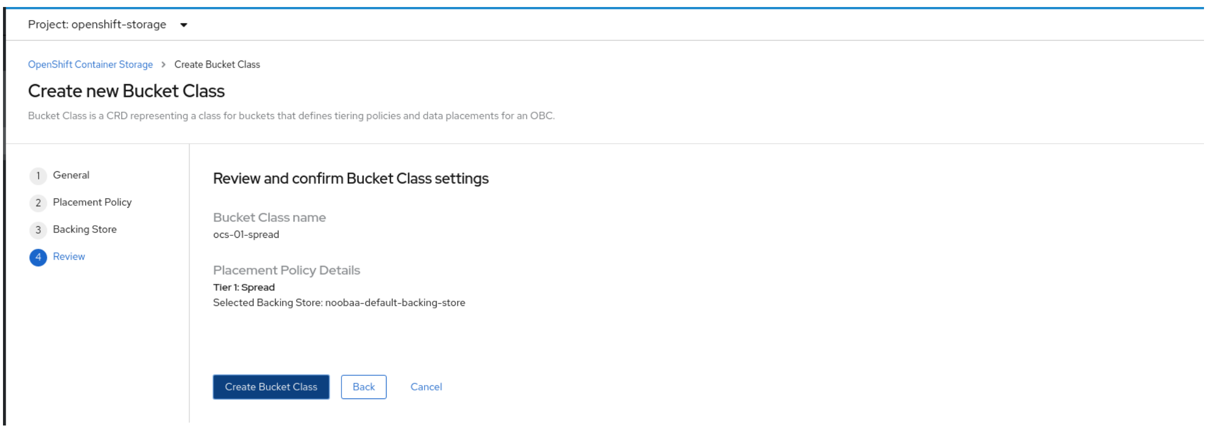


NOTE

You need to select atleast 2 backing stores when you select Policy Type as Mirror in previous step.

a. Review and confirm Bucket Class settings.

Figure 9.5. Bucket class settings review page



b. Click **Create Bucket Class**.

Verification steps

1. Click **Operators → Installed Operators**.
2. Click **OpenShift Container Storage** Operator.
3. Search for the new Bucket Class or click **Bucket Class** tab to view all the Bucket Classes.

9.3.5. Creating a new backing store

Use this procedure to create a new backing store in OpenShift Container Storage.

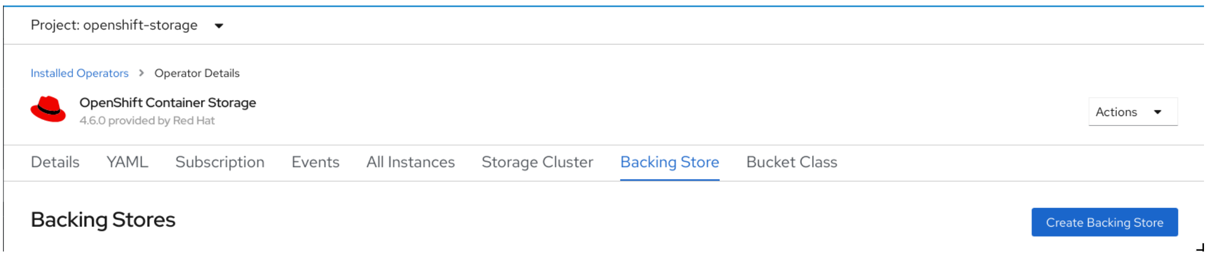
Prerequisites

- Administrator access to OpenShift.

Procedure

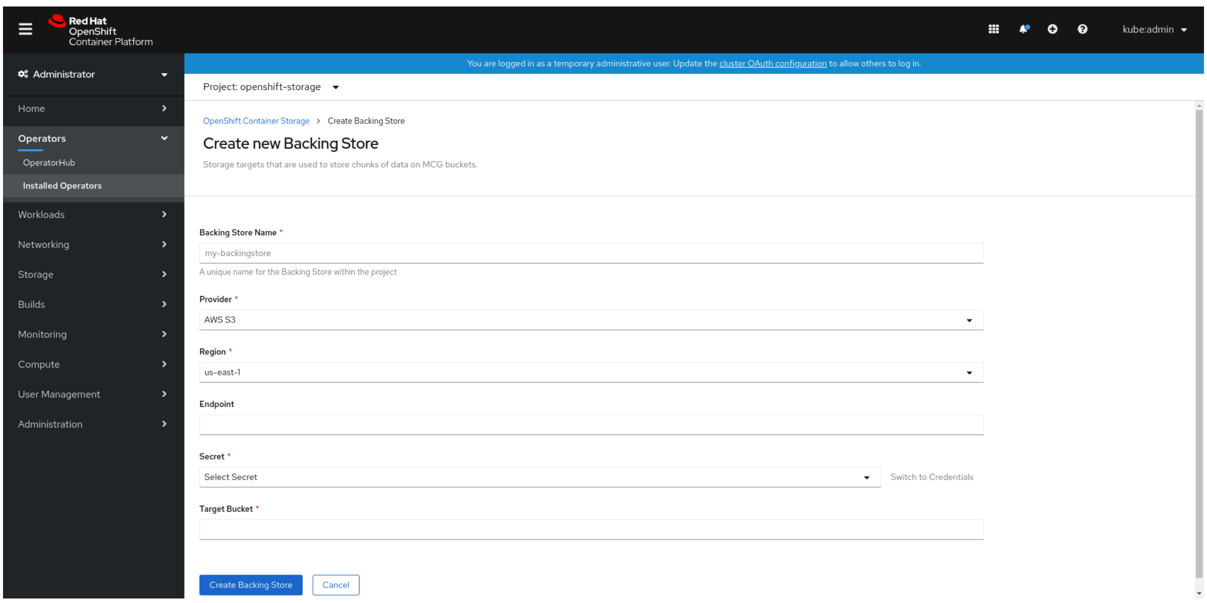
1. Click **Operators → Installed Operators** from the left pane of the OpenShift Web Console to view the installed operators.
2. Click **OpenShift Container Storage** Operator.
3. On the OpenShift Container Storage Operator page, scroll right and click the **Backing Store** tab.

Figure 9.6. OpenShift Container Storage Operator page with backing store tab



4. Click **Create Backing Store**.

Figure 9.7. Create Backing Store page



5. On the Create New Backing Store page, perform the following:
    a. Enter a **Backing Store Name**.
    b. Select a **Provider**.
    c. Select a **Region**.
    d. Enter an **Endpoint**. This is optional.
    e. Select a **Secret** from drop down list, or create your own secret. Optionally, you can **Switch to Credentials** view which lets you fill in the required secrets.
    For more information on creating an OCP secret, see the section Creating the secret in the Openshift Container Platform documentation.

    Each backingstore requires a different secret. For more information on creating the secret for a particular backingstore, see the Section 9.3.1, "Adding storage resources for hybrid or Multicloud using the MCG command line interface" and follow the procedure for the addition of storage resources using a YAML.

    NOTE

    This menu is relevant for all providers except Google Cloud and local PVC.

    f. Enter **Target bucket**. The target bucket is a container storage that is hosted on the remote cloud service. It allows you to create a connection that tells MCG that it can use this bucket for the system.
6. Click **Create Backing Store**.

Verification steps

1. Click **Operators → Installed Operators**.
2. Click **OpenShift Container Storage** Operator.
3. Search for the new backing store or click **Backing Store** tab to view all the backing stores.

9.4. MIRRORING DATA FOR HYBRID AND MULTICLOUD BUCKETS

The Multicloud Object Gateway (MCG) simplifies the process of spanning data across cloud provider and clusters.

Prerequisites

- You must first add a backing storage that can be used by the MCG, see Section 9.3, "Adding storage resources for hybrid or Multicloud".

Then you create a bucket class that reflects the data management policy, mirroring.

Procedure

You can set up mirroring data three ways:

- Section 9.4.1, "Creating bucket classes to mirror data using the MCG command-line-interface"

- Section 9.4.2, "Creating bucket classes to mirror data using a YAML"
- Section 9.4.3, "Configuring buckets to mirror data using the user interface"

9.4.1. Creating bucket classes to mirror data using the MCG command-line-interface

1. From the MCG command-line interface, run the following command to create a bucket class with a mirroring policy:

```
$ noobaa bucketclass create mirror-to-aws --backingstores=azure-resource,aws-resource --placement Mirror
```

2. Set the newly created bucket class to a new bucket claim, generating a new bucket that will be mirrored between two locations:

```
$ noobaa obc create  mirrored-bucket --bucketclass=mirror-to-aws
```

9.4.2. Creating bucket classes to mirror data using a YAML

1. Apply the following YAML. This YAML is a hybrid example that mirrors data between local Ceph storage and AWS:

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  name: hybrid-class
  labels:
    app: noobaa
spec:
  placementPolicy:
    tiers:
      - tier:
          mirrors:
            - mirror:
                spread:
                  - cos-east-us
            - mirror:
                spread:
                  - noobaa-test-bucket-for-ocp201907291921-11247_resource
```

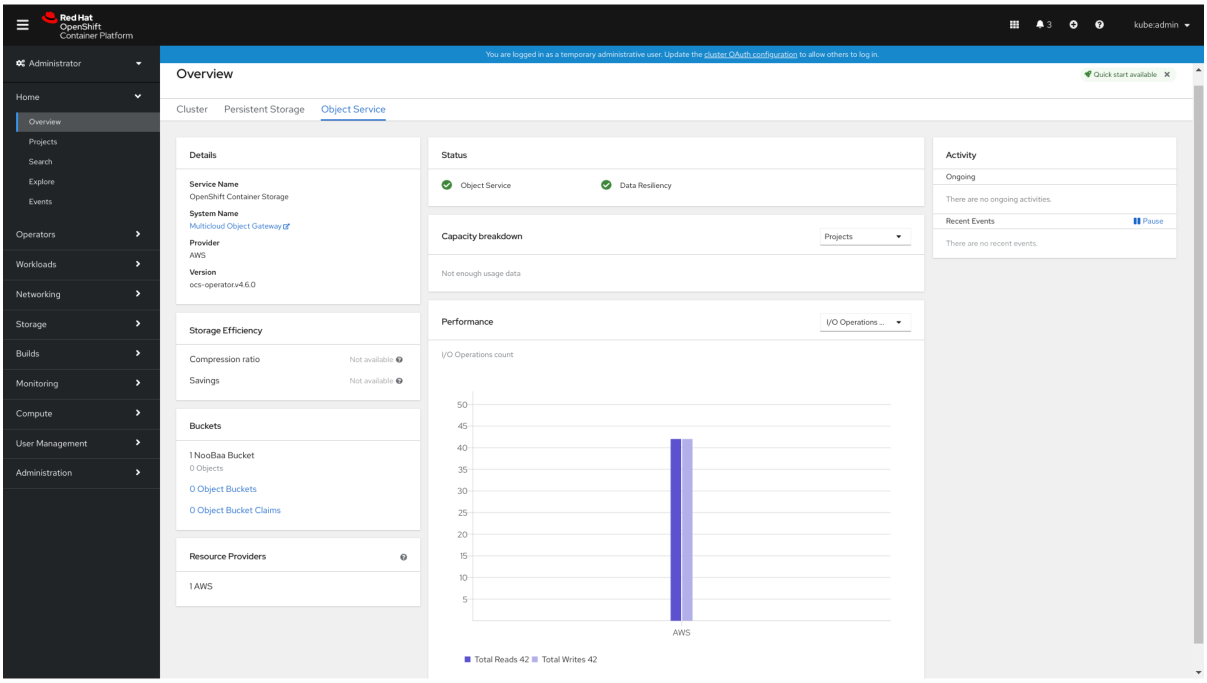
2. Add the following lines to your standard Object Bucket Claim (OBC):

```
additionalConfig:
  bucketclass: mirror-to-aws
```

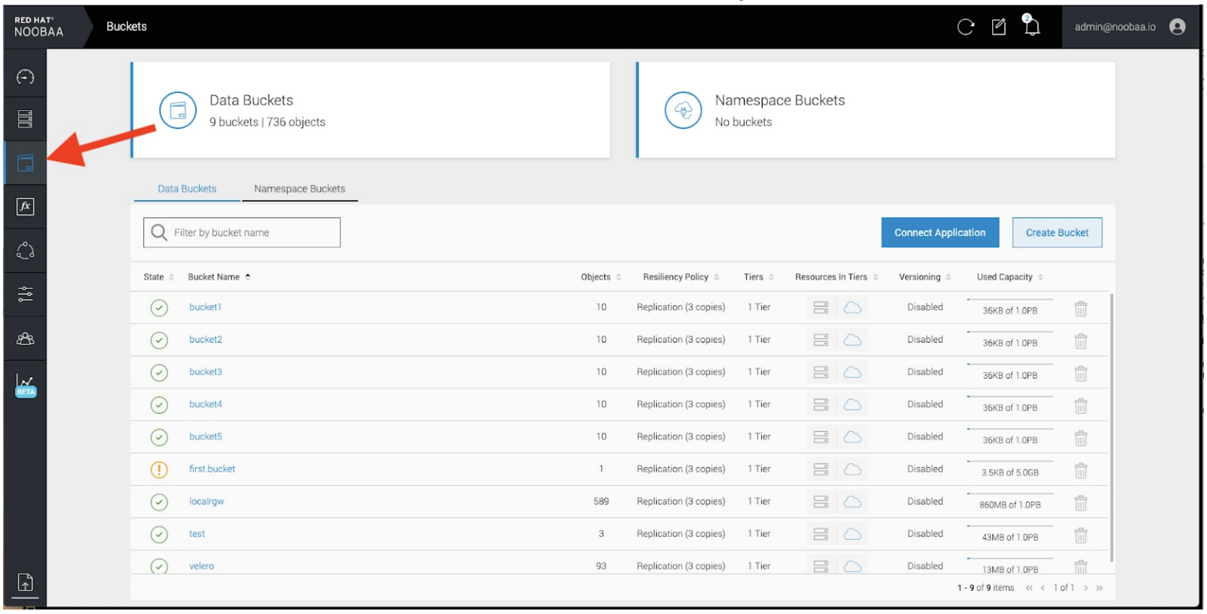
For more information about OBCs, see Section 9.6, "Object Bucket Claim" .

9.4.3. Configuring buckets to mirror data using the user interface

1. In your OpenShift Storage console, navigate to **Overview** → **Object Service** → select the **Multicloud Object Gateway** link:

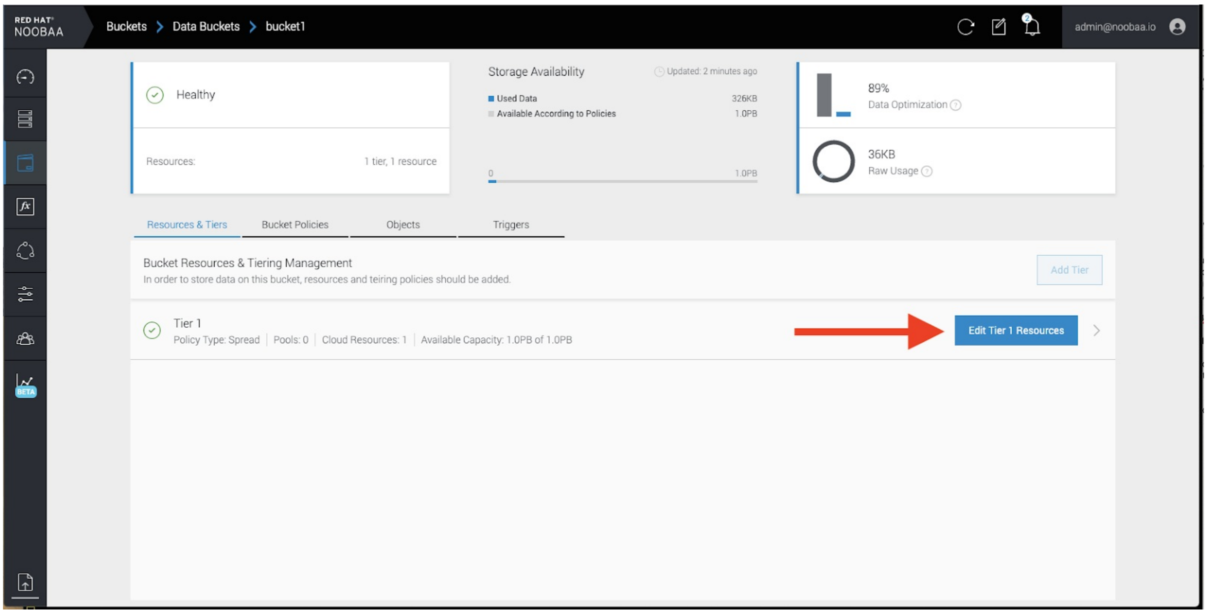


2. Click the **buckets** icon on the left side. You will see a list of your buckets:

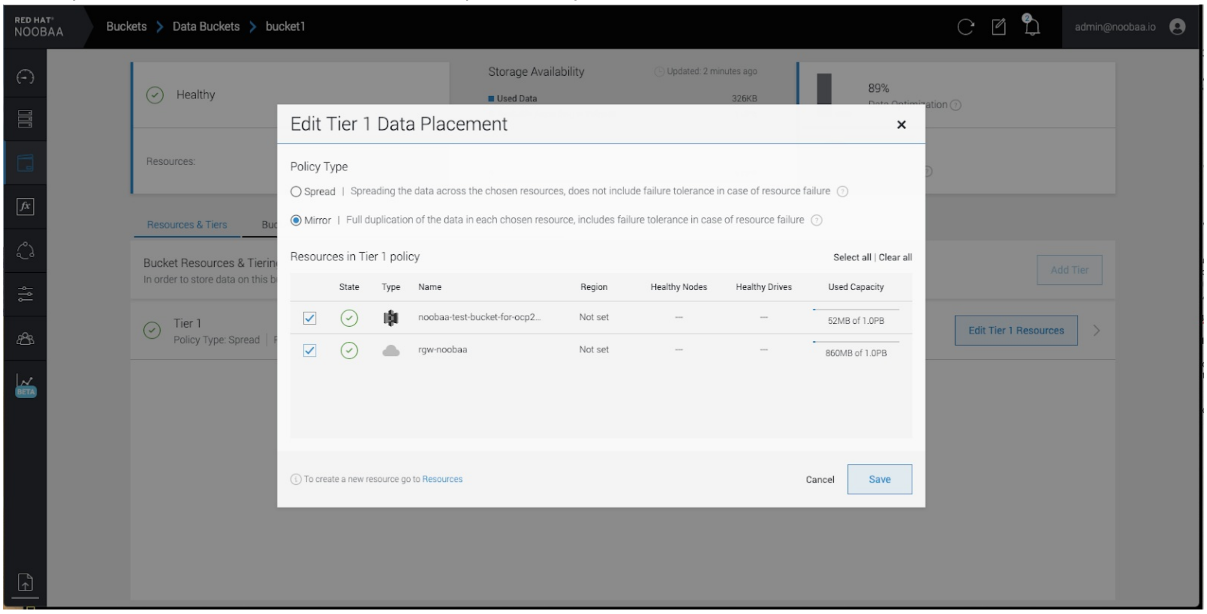


3. Click the bucket you want to update.

4. Click **Edit Tier 1 Resources**:



5. Select **Mirror** and check the relevant resources you want to use for this bucket. In the following example, we mirror data between on prem Ceph RGW to AWS:



6. Click **Save**.

NOTE

Resources created in NooBaa UI cannot be used by OpenShift UI or MCG CLI.

9.5. BUCKET POLICIES IN THE MULTICLOUD OBJECT GATEWAY

OpenShift Container Storage supports AWS S3 bucket policies. Bucket policies allow you to grant users access permissions for buckets and the objects in them.

9.5.1. About bucket policies

Bucket policies are an access policy option available for you to grant permission to your AWS S3 buckets and objects. Bucket policies use JSON-based access policy language. For more information about access policy language, see AWS Access Policy Language Overview .

9.5.2. Using bucket policies

Prerequisites

- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see Section 9.2, “Accessing the Multicloud Object Gateway with your applications”

Procedure

To use bucket policies in the Multicloud Object Gateway:

1. Create the bucket policy in JSON format. See the following example:

```
{
    "Version": "NewVersion",
    "Statement": [
        {
            "Sid": "Example",
            "Effect": "Allow",
            "Principal": [
                    "john.doe@example.com"
            ],
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::john_bucket"
            ]
        }
    ]
}
```

   There are many available elements for bucket policies. For details on these elements and examples of how they can be used, see AWS Access Policy Language Overview .

   For more examples of bucket policies, see AWS Bucket Policy Examples .

   Instructions for creating S3 users can be found in Section 9.5.3, “Creating an AWS S3 user in the Multicloud Object Gateway”.

2. Using AWS S3 client, use the **put-bucket-policy** command to apply the bucket policy to your S3 bucket:

```
# aws --endpoint ENDPOINT --no-verify-ssl s3api put-bucket-policy --bucket MyBucket --
policy BucketPolicy
```

   Replace ***ENDPOINT*** with the S3 endpoint

   Replace ***MyBucket*** with the bucket to set the policy on

   Replace ***BucketPolicy*** with the bucket policy JSON file

   Add **--no-verify-ssl** if you are using the default self signed certificates

For example:

```
# aws --endpoint https://s3-openshift-storage.apps.gogo44.noobaa.org --no-verify-ssl s3api put-bucket-policy -bucket MyBucket --policy file://BucketPolicy
```

For more information on the **put-bucket-policy** command, see the AWS CLI Command Reference for put-bucket-policy.

> **NOTE**
>
> The principal element specifies the user that is allowed or denied access to a resource, such as a bucket. Currently, Only NooBaa accounts can be used as principals. In the case of object bucket claims, NooBaa automatically create an account **obc-account.<generated bucket name>@noobaa.io**.

> **NOTE**
>
> Bucket policy conditions are not supported.

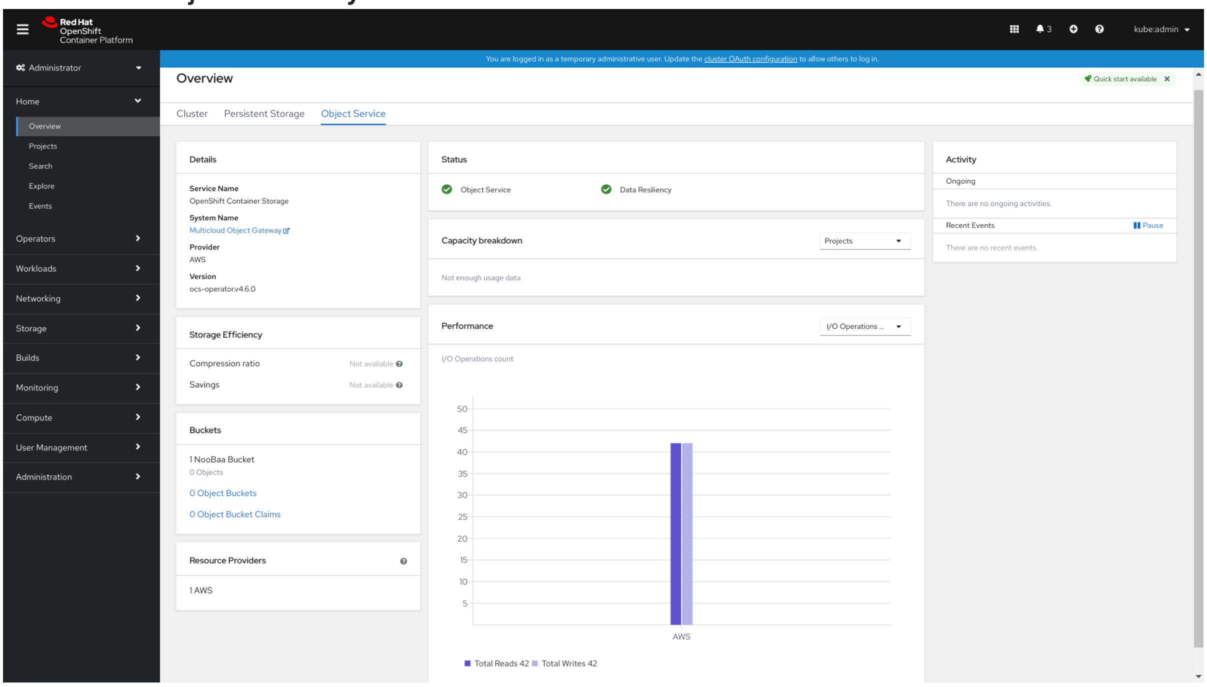
9.5.3. Creating an AWS S3 user in the Multicloud Object Gateway

Prerequisites

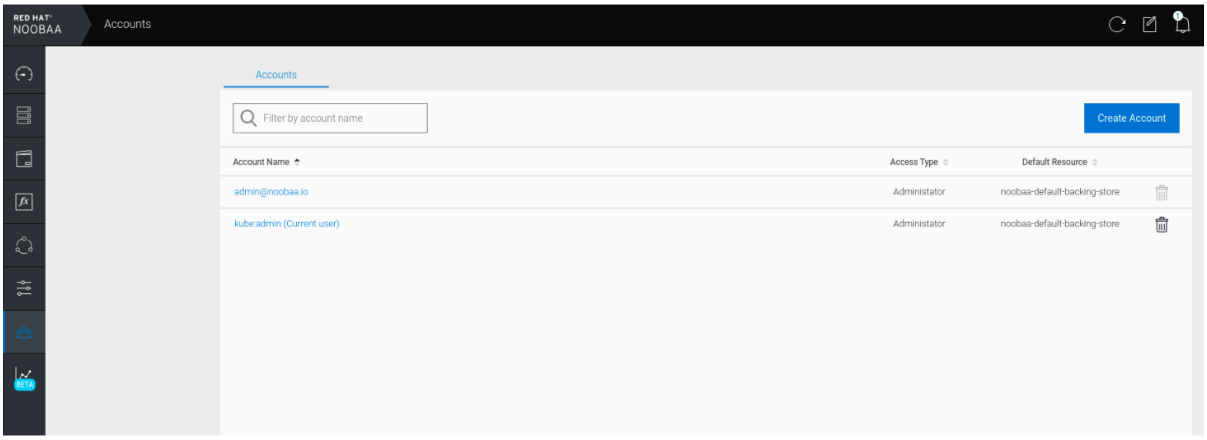
- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see Section 9.2, “Accessing the Multicloud Object Gateway with your applications”

Procedure

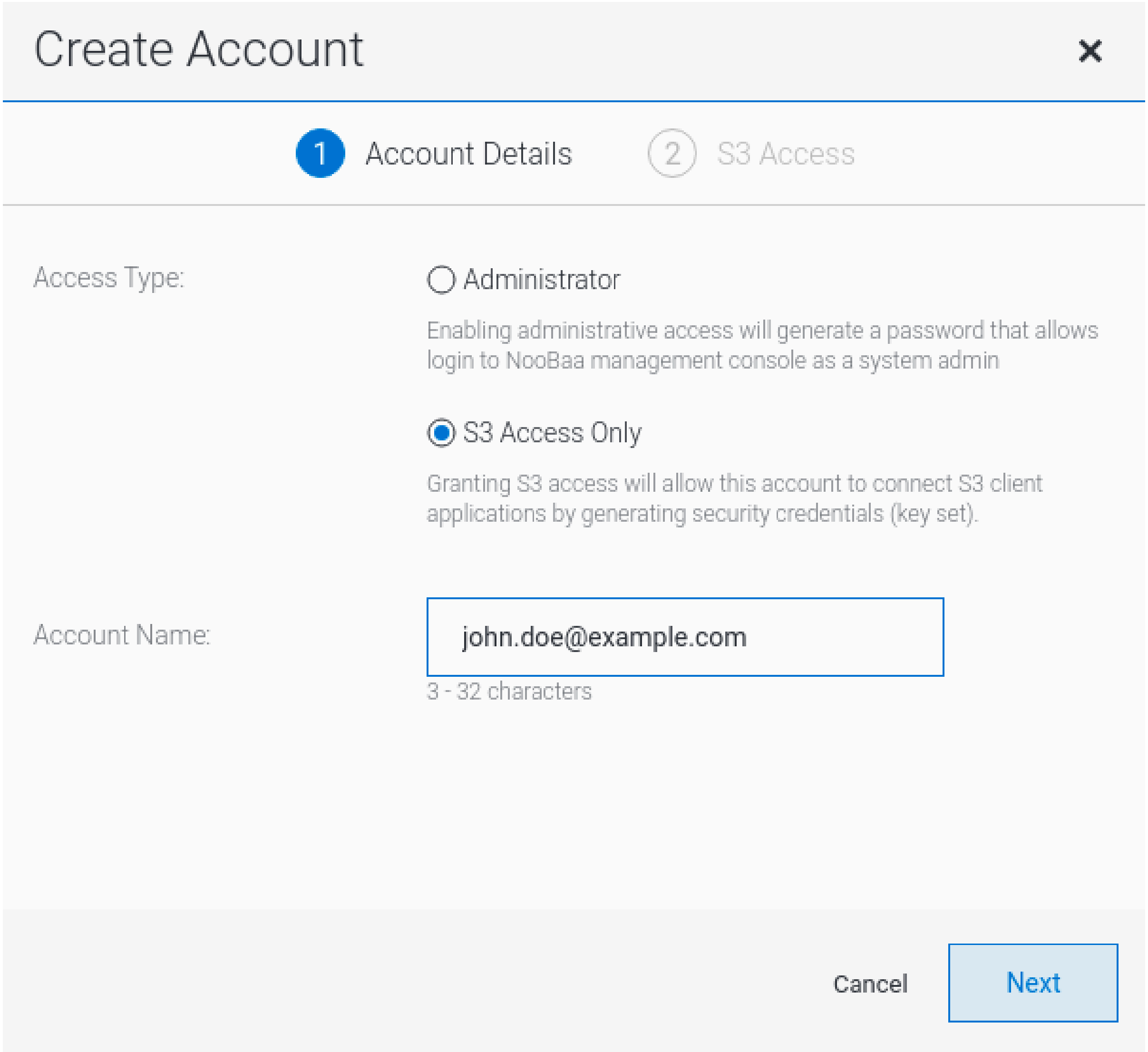
1. In your OpenShift Storage console, navigate to **Overview** → **Object Service** → select the **Multicloud Object Gateway** link:



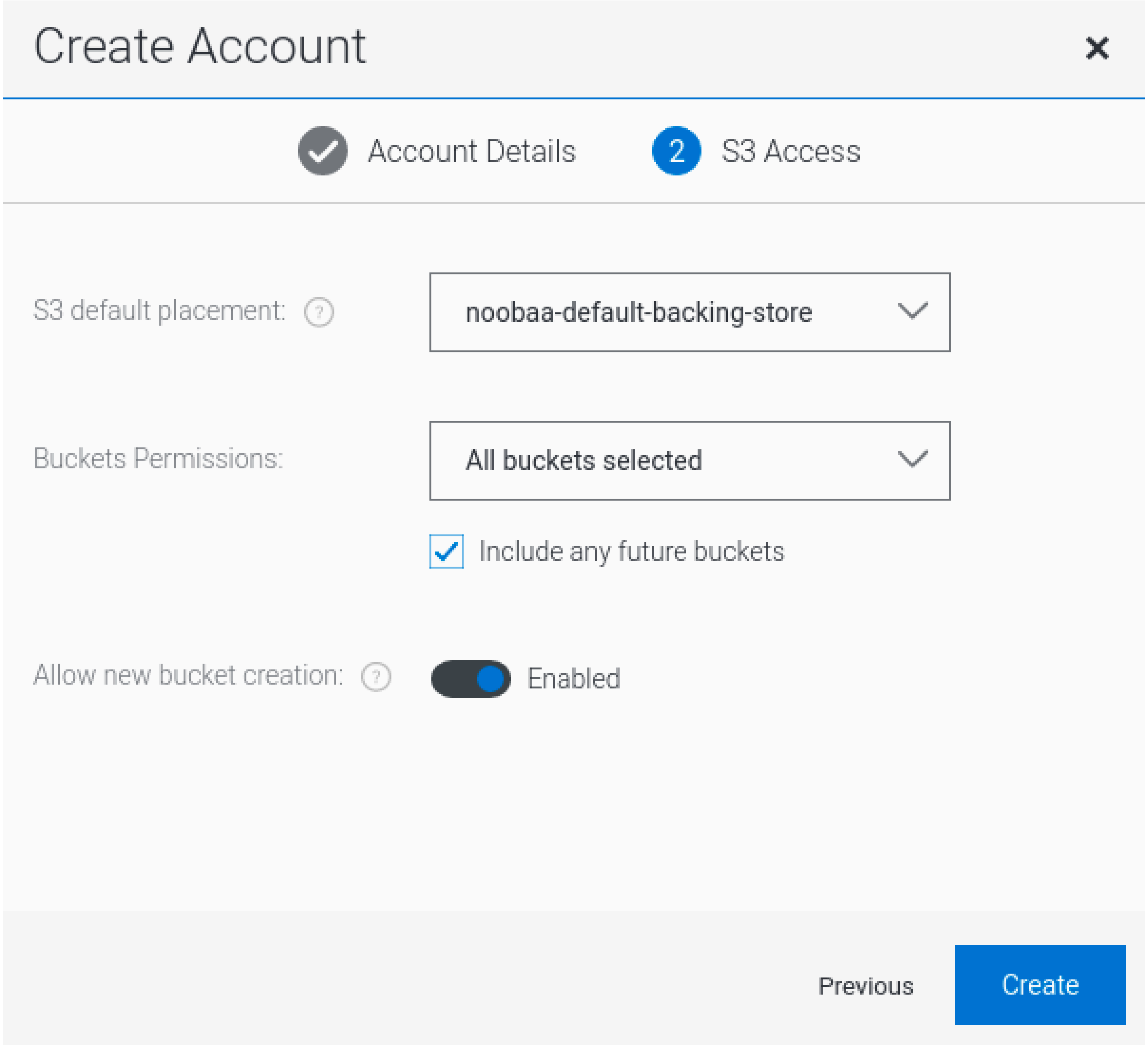
2. Under the **Accounts** tab, click **Create Account**



3. Select **S3 Access Only**, provide the **Account Name**, for example, john.doe@example.com. Click Next:



4. Select **S3 default placement**, for example, noobaa-default-backing-store. Select **Buckets Permissions**. A specific bucket or all buckets can be selected. Click **Create**:



9.6. OBJECT BUCKET CLAIM

An Object Bucket Claim can be used to request an S3 compatible bucket backend for your workloads.

You can create an Object Bucket Claim three ways:

- Section 9.6.1, "Dynamic Object Bucket Claim"
- Section 9.6.2, "Creating an Object Bucket Claim using the command line interface"
- Section 9.6.3, "Creating an Object Bucket Claim using the OpenShift Web Console"

An object bucket claim creates a new bucket and an application account in NooBaa with permissions to the bucket, including a new access key and secret access key. The application account is allowed to access only a single bucket and can't create new buckets by default.

9.6.1. Dynamic Object Bucket Claim

Similar to Persistent Volumes, you can add the details of the Object Bucket claim to your application's YAML, and get the object service endpoint, access key, and secret access key available in a configuration map and secret. It is easy to read this information dynamically into environment variables of your application.

Procedure

1. Add the following lines to your application YAML:

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <obc-name>
spec:
  generateBucketName: <obc-bucket-name>
  storageClassName: openshift-storage.noobaa.io
```

These lines are the Object Bucket Claim itself.

a. Replace **<obc-name>** with the a unique Object Bucket Claim name.

b. Replace **<obc-bucket-name>** with a unique bucket name for your Object Bucket Claim.

2. You can add more lines to the YAML file to automate the use of the Object Bucket Claim. The example below is the mapping between the bucket claim result, which is a configuration map with data and a secret with the credentials. This specific job will claim the Object Bucket from NooBaa, which will create a bucket and an account.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: testjob
spec:
  template:
    spec:
      restartPolicy: OnFailure
      containers:
        - image: <your application image>
          name: test
          env:
            - name: BUCKET_NAME
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_NAME
            - name: BUCKET_HOST
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_HOST
            - name: BUCKET_PORT
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_PORT
            - name: AWS_ACCESS_KEY_ID
              valueFrom:
                secretKeyRef:
                  name: <obc-name>
                  key: AWS_ACCESS_KEY_ID
            - name: AWS_SECRET_ACCESS_KEY
```

```
          valueFrom:
            secretKeyRef:
              name: <obc-name>
              key: AWS_SECRET_ACCESS_KEY
```

a. Replace all instances of <obc-name> with your Object Bucket Claim name.

b. Replace <your application image> with your application image.

3. Apply the updated YAML file:

```
# oc apply -f <yaml.file>
```

a. Replace **<yaml.file>** with the name of your YAML file.

4. To view the new configuration map, run the following:

```
# oc get cm <obc-name>
```

a. Replace **obc-name** with the name of your Object Bucket Claim.
   You can expect the following environment variables in the output:

   - **BUCKET_HOST** - Endpoint to use in the application
   - **BUCKET_PORT** - The port available for the application
     - The port is related to the **BUCKET_HOST**. For example, if the **BUCKET_HOST** is https://my.example.com, and the **BUCKET_PORT** is 443, the endpoint for the object service would be https://my.example.com:443.
   - **BUCKET_NAME** - Requested or generated bucket name
   - **AWS_ACCESS_KEY_ID** - Access key that is part of the credentials
   - **AWS_SECRET_ACCESS_KEY** - Secret access key that is part of the credentials

9.6.2. Creating an Object Bucket Claim using the command line interface

When creating an Object Bucket Claim using the command-line interface, you get a configuration map and a Secret that together contain all the information your application needs to use the object storage service.

Prerequisites

- Download the MCG command-line interface:

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

Procedure

1. Use the command-line interface to generate the details of a new bucket and credentials. Run the following command:

```
# noobaa obc create <obc-name> -n openshift-storage
```

–

Replace **<obc-name>** with a unique Object Bucket Claim name, for example, **myappobc**.

Additionally, you can use the **--app-namespace** option to specify the namespace where the Object Bucket Claim configuration map and secret will be created, for example, **myapp-namespace**.

Example output:

```
INFO[0001]   Created: ObjectBucketClaim "test21obc"
```

The MCG command-line-interface has created the necessary configuration and has informed OpenShift about the new OBC.

2. Run the following command to view the Object Bucket Claim:

```
# oc get obc -n openshift-storage
```

Example output:

```
NAME        STORAGE-CLASS                 PHASE   AGE
test21obc   openshift-storage.noobaa.io   Bound   38s
```

3. Run the following command to view the YAML file for the new Object Bucket Claim:

```
# oc get obc test21obc -o yaml -n openshift-storage
```

Example output:


```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  generation: 2
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  resourceVersion: "40756"
  selfLink: /apis/objectbucket.io/v1alpha1/namespaces/openshift-
storage/objectbucketclaims/test21obc
  uid: 64f04cba-f662-11e9-bc3c-0295250841af
spec:
  ObjectBucketName: obc-openshift-storage-test21obc
  bucketName: test21obc-933348a6-e267-4f82-82f1-e59bf4fe3bb4
  generateBucketName: test21obc
  storageClassName: openshift-storage.noobaa.io
status:
  phase: Bound
```

4. Inside of your **openshift-storage** namespace, you can find the configuration map and the secret to use this Object Bucket Claim. The CM and the secret have the same name as the Object Bucket Claim. To view the secret:

```
# oc get -n openshift-storage secret test21obc -o yaml
```

Example output:


```
Example output:
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: c0M0R2xVanF3ODR3bHBkVW94cmY=
  AWS_SECRET_ACCESS_KEY:
Wi9kcFluSWxHRzlWaFlzNk1hc0xma2JXcjM1MVhqa051SlBleXpmOQ==
kind: Secret
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  ownerReferences:
  - apiVersion: objectbucket.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: ObjectBucketClaim
    name: test21obc
    uid: 64f04cba-f662-11e9-bc3c-0295250841af
  resourceVersion: "40751"
  selfLink: /api/v1/namespaces/openshift-storage/secrets/test21obc
  uid: 65117c1c-f662-11e9-9094-0a5305de57bb
type: Opaque
```


The secret gives you the S3 access credentials.

5. To view the configuration map:

```
# oc get -n openshift-storage cm test21obc -o yaml
```

Example output:


```
apiVersion: v1
data:
  BUCKET_HOST: 10.0.171.35
  BUCKET_NAME: test21obc-933348a6-e267-4f82-82f1-e59bf4fe3bb4
  BUCKET_PORT: "31242"
  BUCKET_REGION: ""
  BUCKET_SUBREGION: ""
kind: ConfigMap
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
```

```
finalizers:
- objectbucket.io/finalizer
labels:
  app: noobaa
  bucket-provisioner: openshift-storage.noobaa.io-obc
  noobaa-domain: openshift-storage.noobaa.io
name: test21obc
namespace: openshift-storage
ownerReferences:
- apiVersion: objectbucket.io/v1alpha1
  blockOwnerDeletion: true
  controller: true
  kind: ObjectBucketClaim
  name: test21obc
  uid: 64f04cba-f662-11e9-bc3c-0295250841af
resourceVersion: "40752"
selfLink: /api/v1/namespaces/openshift-storage/configmaps/test21obc
uid: 651c6501-f662-11e9-9094-0a5305de57bb
```

The configuration map contains the S3 endpoint information for your application.

9.6.3. Creating an Object Bucket Claim using the OpenShift Web Console

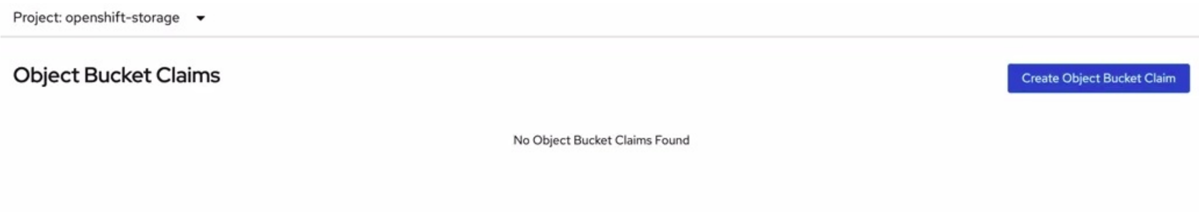
You can create an Object Bucket Claim (OBC) using the OpenShift Web Console.

Prerequisites

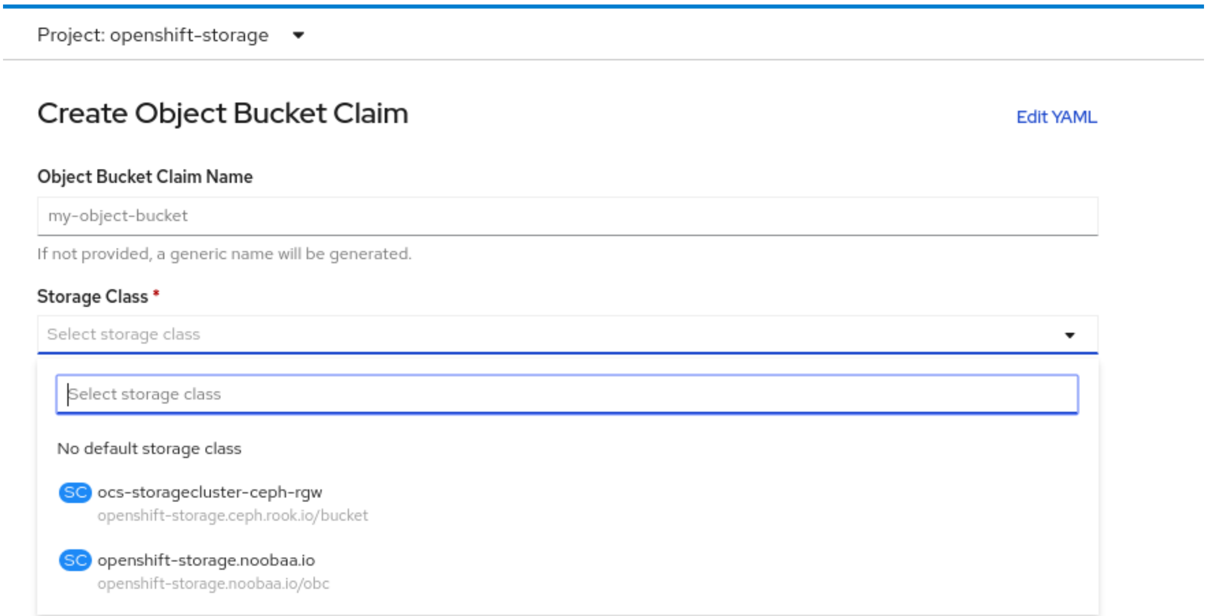
- Administrative access to the OpenShift Web Console.
- In order for your applications to communicate with the OBC, you need to use the configmap and secret. For more information about this, see Section 9.6.1, “Dynamic Object Bucket Claim” .

Procedure

1. Log into the OpenShift Web Console.
2. On the left navigation bar, click **Storage → Object Bucket Claims**.
3. Click **Create Object Bucket Claim**.



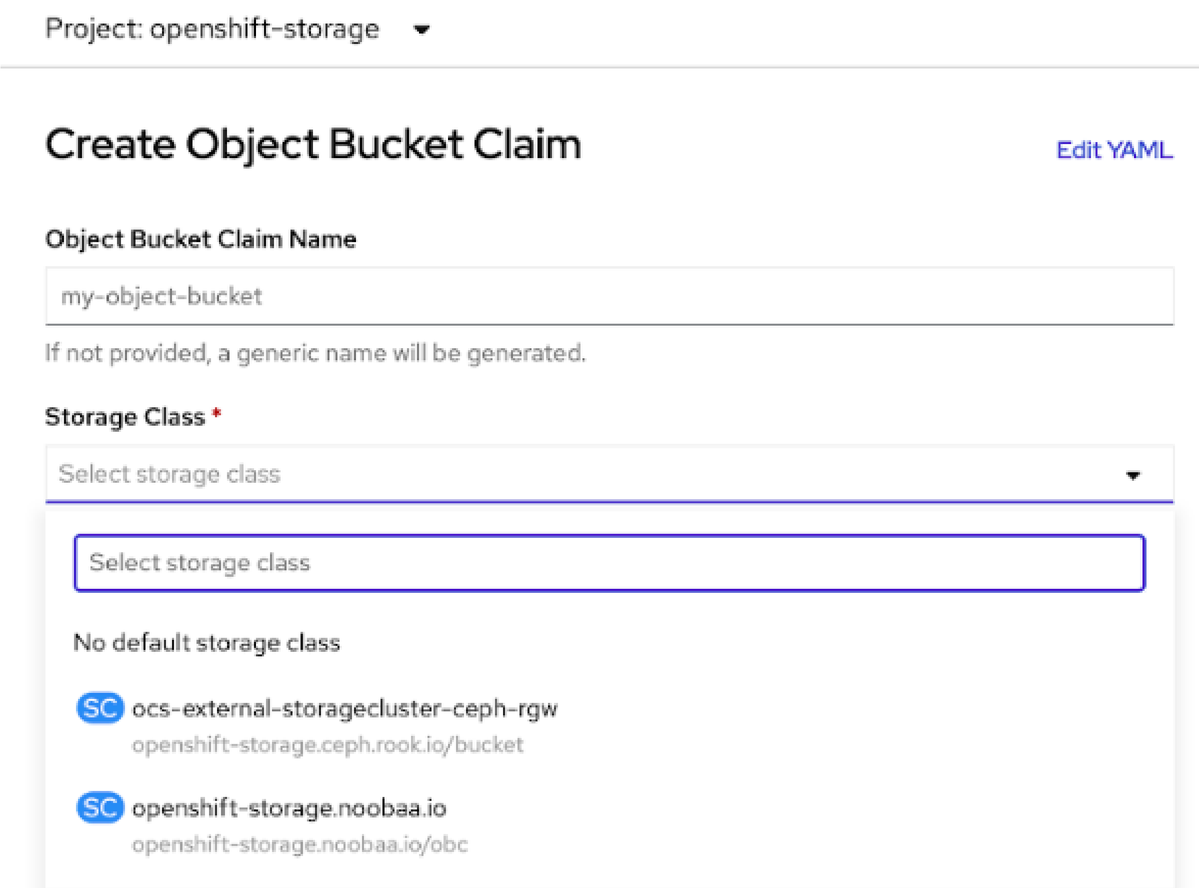
4. Enter a name for your object bucket claim and select the appropriate storage class based on your deployment, internal or external, from the dropdown menu:
   Internal mode



The following storage classes, which were created after deployment, are available for use:

- **ocs-storagecluster-ceph-rgw** uses the Ceph Object Gateway (RGW)
- **openshift-storage.noobaa.io** uses the Multicloud Object Gateway

External mode



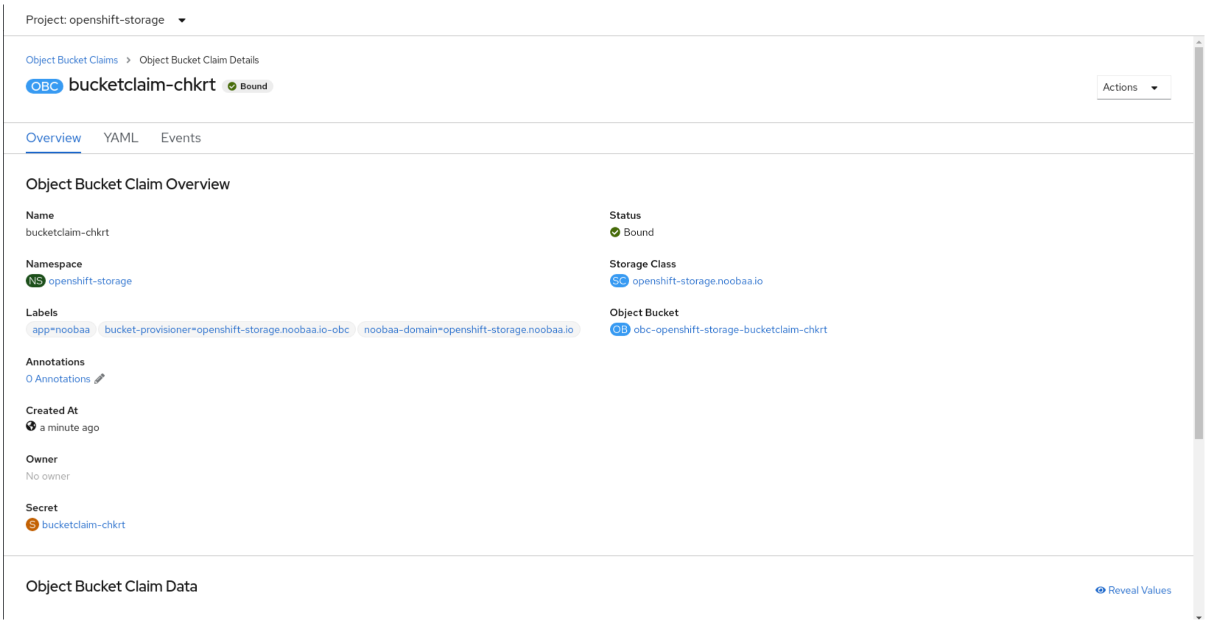
The following storage classes, which were created after deployment, are available for use:

- **ocs-external-storagecluster-ceph-rgw** uses the Ceph Object Gateway (RGW)

- **openshift-storage.noobaa.io** uses the Multicloud Object Gateway

> **NOTE**
>
> The RGW OBC storage class is only available with fresh installations of OpenShift Container Storage version 4.5. It does not apply to clusters upgraded from previous OpenShift Container Storage releases.

5. Click **Create**.
   Once you create the OBC, you are redirected to its detail page:



Additional Resources

- Section 9.6, "Object Bucket Claim"

9.7. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS

The Multicloud Object Gateway performance may vary from one environment to another. In some cases, specific applications require faster performance which can be easily addressed by scaling S3 endpoints.

The Multicloud Object Gateway resource pool is a group of NooBaa daemon containers that provide two types of services enabled by default:

- Storage service
- S3 endpoint service

9.7.1. S3 endpoints in the Multicloud Object Gateway

The S3 endpoint is a service that every Multicloud Object Gateway provides by default that handles the heavy lifting data digestion in the Multicloud Object Gateway. The endpoint service handles the inline data chunking, deduplication, compression, and encryption, and it accepts data placement instructions from the Multicloud Object Gateway.

9.7.2. Scaling with storage nodes

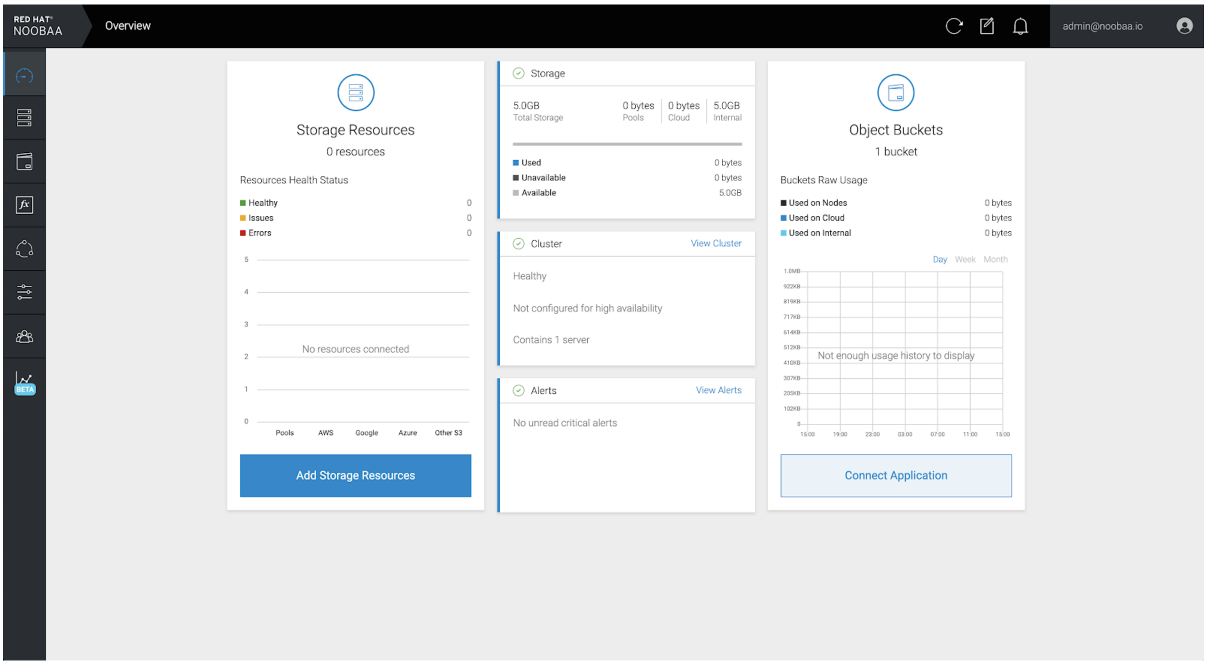
Prerequisites

- A running OpenShift Container Storage cluster on OpenShift Container Platform with access to the Multicloud Object Gateway.

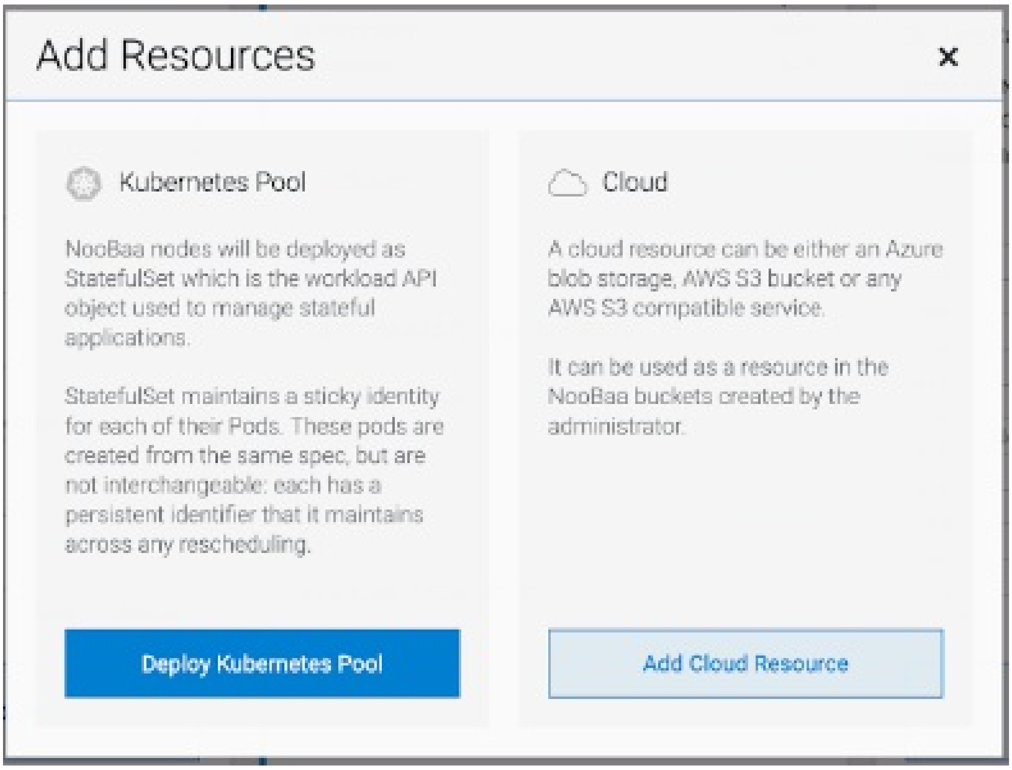
A storage node in the Multicloud Object Gateway is a NooBaa daemon container attached to one or more Persistent Volumes and used for local object service data storage. NooBaa daemons can be deployed on Kubernetes nodes. This can be done by creating a Kubernetes pool consisting of StatefulSet pods.

Procedure

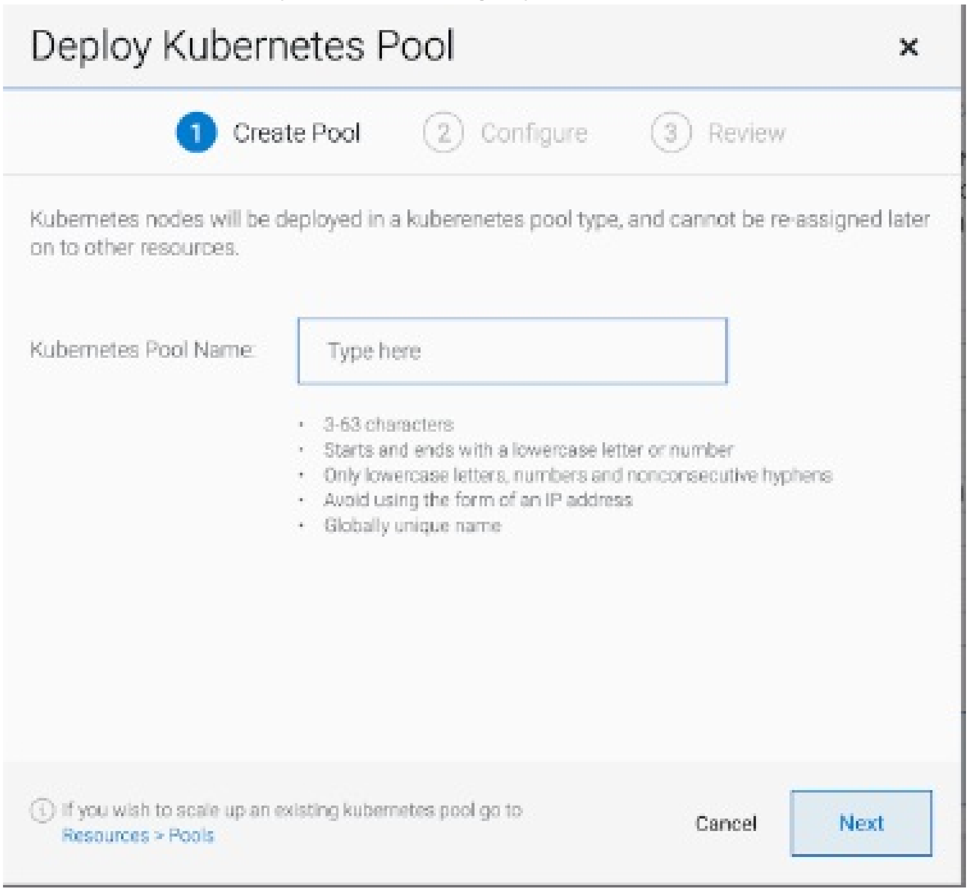
1. In the Multicloud Object Gateway user interface, from the **Overview** page, click **Add Storage Resources**:



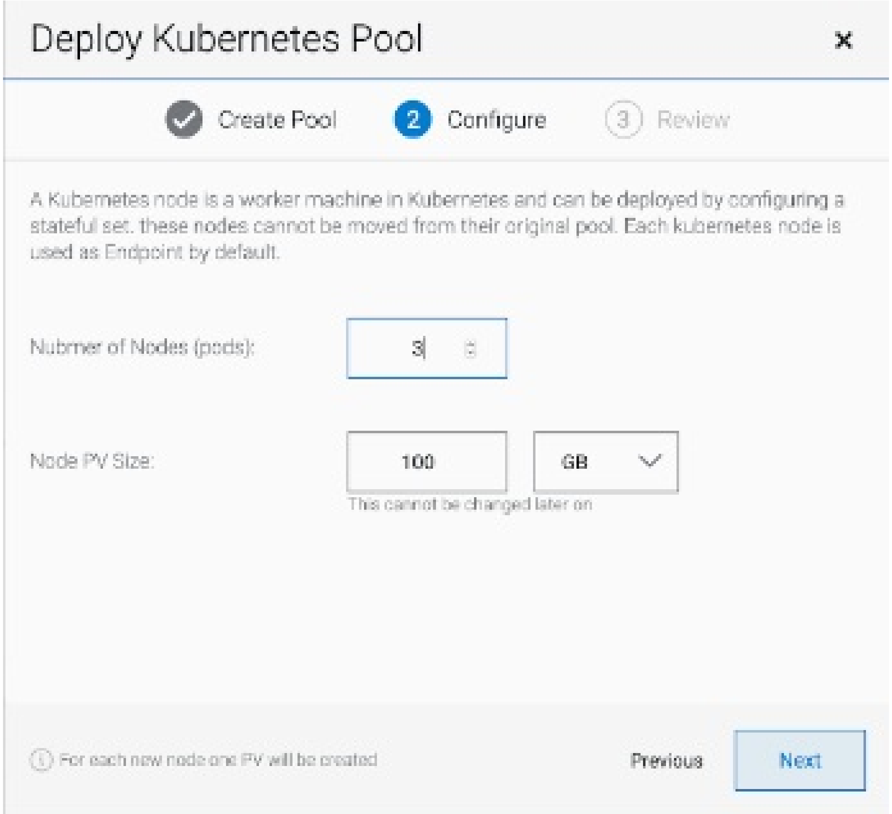
2. In the window, click **Deploy Kubernetes Pool**



3. In the **Create Pool** step create the target pool for the future installed nodes.

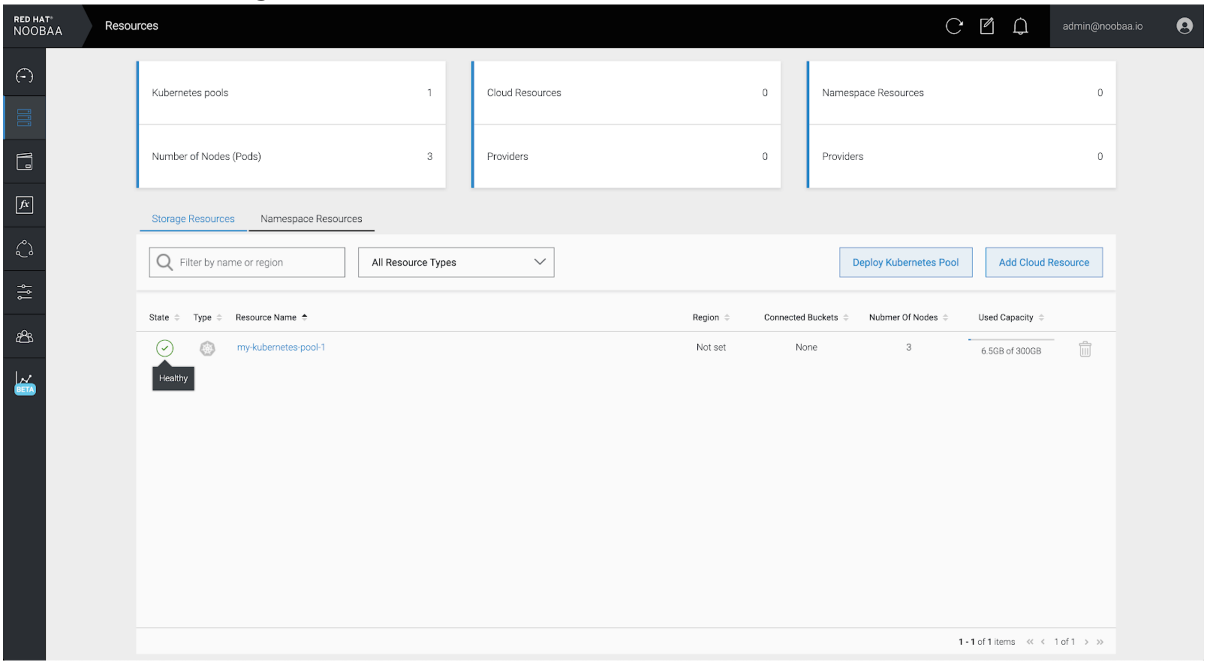


4. In the **Configure** step, configure the number of requested pods and the size of each PV. For each new pod, one PV is be created.



5. In the **Review** step, you can find the details of the new pool and select the deployment method you wish to use: local or external deployment. If local deployment is selected, the Kubernetes nodes will deploy within the cluster. If external deployment is selected, you will be provided with a YAML file to run externally.

6. All nodes will be assigned to the pool you chose in the first step, and can be found under **Resources → Storage resources→ Resource name**:



CHAPTER 10. MANAGING PERSISTENT VOLUME CLAIMS

10.1. CONFIGURING APPLICATION PODS TO USE OPENSHIFT CONTAINER STORAGE

Follow the instructions in this section to configure OpenShift Container Storage as storage for an application pod.

Prerequisites

- You have administrative access to OpenShift Web Console.
- OpenShift Container Storage Operator is installed and running in the **openshift-storage** namespace. In OpenShift Web Console, click **Operators → Installed Operators** to view installed operators.
- The default storage classes provided by OpenShift Container Storage are available. In OpenShift Web Console, click **Storage → Storage Classes** to view default storage classes.

Procedure

1. **Create a Persistent Volume Claim (PVC) for the application to use.**
    a. In OpenShift Web Console, click **Storage → Persistent Volume Claims**.
    b. Set the **Project** for the application pod.
    c. Click **Create Persistent Volume Claim**
        i. Specify a **Storage Class** provided by OpenShift Container Storage.
        ii. Specify the PVC **Name**, for example, **myclaim**.
        iii. Select the required **Access Mode**.
        iv. Specify a **Size** as per application requirement.
        v. Click **Create** and wait until the PVC is in **Bound** status.
2. **Configure a new or existing application pod to use the new PVC.**
    - For a new application pod, perform the following steps:
        i. Click **Workloads →Pods**.
        ii. Create a new application pod.
        iii. Under the **spec:** section, add **volume:** section to add the new PVC as a volume for the application pod.

```
volumes:
  - name: <volume_name>
    persistentVolumeClaim:
      claimName: <pvc_name>
```

For example:

```
volumes:
  - name: mypd
    persistentVolumeClaim:
      claimName: myclaim
```

- For an existing application pod, perform the following steps:
    i. Click **Workloads →Deployment Configs**.
    ii. Search for the required deployment config associated with the application pod.
    iii. Click on its **Action menu (⋮) → Edit Deployment Config**.
    iv. Under the **spec:** section, add **volume:** section to add the new PVC as a volume for the application pod and click **Save**.

```
volumes:
  - name: <volume_name>
    persistentVolumeClaim:
      claimName: <pvc_name>
```

For example:

```
volumes:
  - name: mypd
    persistentVolumeClaim:
      claimName: myclaim
```

3. **Verify that the new configuration is being used.**
    a. Click **Workloads → Pods**.
    b. Set the **Project** for the application pod.
    c. Verify that the application pod appears with a status of **Running**.
    d. Click the application pod name to view pod details.
    e. Scroll down to **Volumes** section and verify that the volume has a **Type** that matches your new Persistent Volume Claim, for example, **myclaim**.

10.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS

Use this procedure to view the status of a PVC request.

Prerequisites

- Administrator access to OpenShift Container Storage.

Procedure

1. Log in to OpenShift Web Console.
2. Click **Storage → Persistent Volume Claims**

3. Search for the required PVC name by using the **Filter** textbox. You can also filter the list of PVCs by Name or Label to narrow down the list

4. Check the **Status** column corresponding to the required PVC.

5. Click the required **Name** to view the PVC details.

10.3. REVIEWING PERSISTENT VOLUME CLAIM REQUEST EVENTS

Use this procedure to review and address Persistent Volume Claim (PVC) request events.

Prerequisites

- Administrator access to OpenShift Web Console.

Procedure

1. Log in to OpenShift Web Console.

2. Click **Home → Overview → Persistent Storage**

3. Locate the **Inventory** card to see the number of PVCs with errors.

4. Click **Storage → Persistent Volume Claims**

5. Search for the required PVC using the **Filter** textbox.

6. Click on the PVC name and navigate to **Events**

7. Address the events as required or as directed.

10.4. EXPANDING PERSISTENT VOLUME CLAIMS

OpenShift Container Storage 4.6 introduces the ability to expand Persistent Volume Claims providing more flexibility in the management of persistent storage resources.

Expansion is supported for the following Persistent Volumes:

- PVC with ReadWriteOnce (RWO) and ReadWriteMany (RWX) access that is based on Ceph File System (CephFS) for volume mode **Filesystem**.

- PVC with ReadWriteOnce (RWO) access that is based on Ceph RADOS Block Devices (RBDs) with volume mode **Filesystem**.

- PVC with ReadWriteOnce (RWO) access that is based on Ceph RADOS Block Devices (RBDs) with volume mode **Block**.

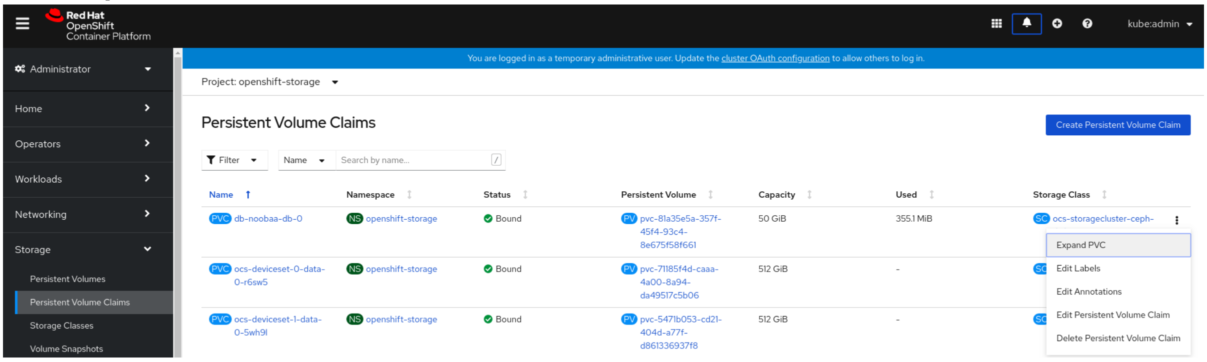
> **WARNING**
>
> OSD and MON PVC expansion is not supported by Red Hat.

Prerequisites

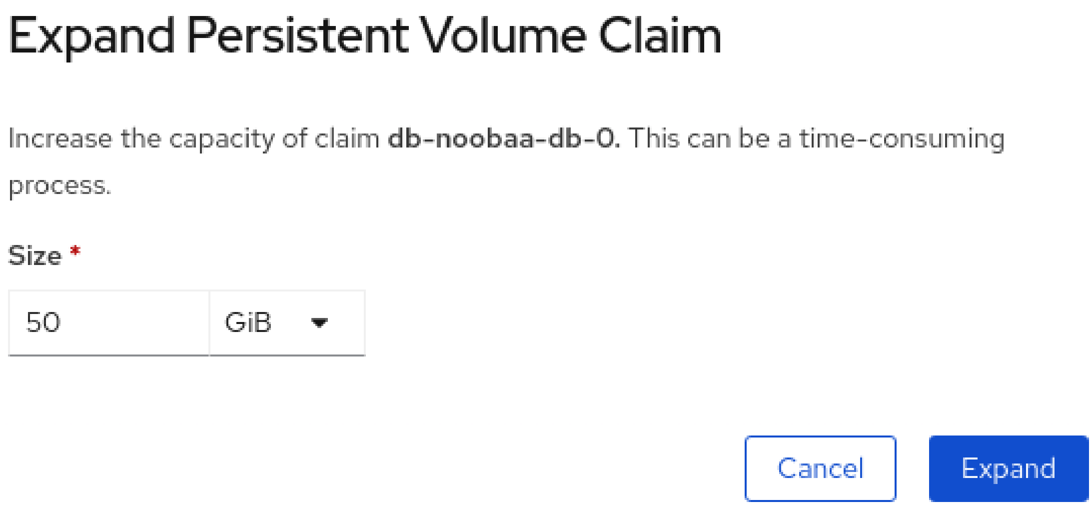
- Administrator access to OpenShift Web Console.

Procedure

1. In OpenShift Web Console, navigate to **Storage → Persistent Volume Claims**.
2. Click the Action Menu (⋮) next to the Persistent Volume Claim you want to expand.
3. Click **Expand PVC**:



4. Select the new size of the Persistent Volume Claim, then click **Expand**:



5. To verify the expansion, navigate to the PVC's details page and verify the **Capacity** field has the correct size requested.

> **NOTE**
>
> When expanding PVCs based on Ceph RADOS Block Devices (RBDs), if the PVC is not already attached to a pod the **Condition type** is **FileSystemResizePending** in the PVC's details page. Once the volume is mounted, filesystem resize succeeds and the new size is reflected in the **Capacity** field.

10.5. DYNAMIC PROVISIONING

10.5.1. About dynamic provisioning

The StorageClass resource object describes and classifies storage that can be requested, as well as provides a means for passing parameters for dynamically provisioned storage on demand. StorageClass objects can also serve as a management mechanism for controlling different levels of storage and access to the storage. Cluster Administrators (**cluster-admin**) or Storage Administrators (**storage-admin**) define and create the StorageClass objects that users can request without needing any intimate knowledge about the underlying storage volume sources.

The OpenShift Container Platform persistent volume framework enables this functionality and allows administrators to provision a cluster with persistent storage. The framework also gives users a way to request those resources without having any knowledge of the underlying infrastructure.

Many storage types are available for use as persistent volumes in OpenShift Container Platform. While all of them can be statically provisioned by an administrator, some types of storage are created dynamically using the built-in provider and plug-in APIs.

10.5.2. Dynamic provisioning in OpenShift Container Storage

Red Hat OpenShift Container Storage is software-defined storage that is optimised for container environments. It runs as an operator on OpenShift Container Platform to provide highly integrated and simplified persistent storage management for containers.

OpenShift Container Storage supports a variety of storage types, including:

- Block storage for databases
- Shared file storage for continuous integration, messaging, and data aggregation
- Object storage for archival, backup, and media storage

Version 4 uses Red Hat Ceph Storage to provide the file, block, and object storage that backs persistent volumes, and Rook.io to manage and orchestrate provisioning of persistent volumes and claims. NooBaa provides object storage, and its Multicloud Gateway allows object federation across multiple cloud environments (available as a Technology Preview).

In OpenShift Container Storage 4, the Red Hat Ceph Storage Container Storage Interface (CSI) driver for RADOS Block Device (RBD) and Ceph File System (CephFS) handles the dynamic provisioning requests. When a PVC request comes in dynamically, the CSI driver has the following options:

- Create a PVC with ReadWriteOnce (RWO) and ReadWriteMany (RWX) access that is based on Ceph RBDs with volume mode **Block**
- Create a PVC with ReadWriteOnce (RWO) access that is based on Ceph RBDs with volume mode **Filesystem**
- Create a PVC with ReadWriteOnce (RWO) and ReadWriteMany (RWX) access that is based on CephFS for volume mode **Filesystem**

The judgment of which driver (RBD or CephFS) to use is based on the entry in the **storageclass.yaml** file.

10.5.3. Available dynamic provisioning plug-ins

OpenShift Container Platform provides the following provisioner plug-ins, which have generic implementations for dynamic provisioning that use the cluster's configured provider's API to create new storage resources:

Storage type	Provisioner plug-in name	Notes
OpenStack Cinder	**kubernetes.io/cinder**	
AWS Elastic Block Store (EBS)	**kubernetes.io/aws-ebs**	For dynamic provisioning when using multiple clusters in different zones, tag each node with **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>** where **<cluster_name>** and **<cluster_id>** are unique per cluster.
AWS Elastic File System (EFS)		Dynamic provisioning is accomplished through the EFS provisioner pod and not through a provisioner plug-in.
Azure Disk	**kubernetes.io/azure-disk**	
Azure File	**kubernetes.io/azure-file**	The **persistent-volume-binder** ServiceAccount requires permissions to create and get Secrets to store the Azure storage account and keys.
GCE Persistent Disk (gcePD)	**kubernetes.io/gce-pd**	In multi-zone configurations, it is advisable to run one OpenShift Container Platform cluster per GCE project to avoid PVs from being created in zones where no node in the current cluster exists.
VMware vSphere	**kubernetes.io/vsphere-volume**	

IMPORTANT

Any chosen provisioner plug-in also requires configuration for the relevant cloud, host, or third-party provider as per the relevant documentation.

CHAPTER 11. VOLUME SNAPSHOTS

A volume snapshot is the state of the storage volume in a cluster at a particular point in time. These snapshots help to use storage more efficiently by not having to make a full copy each time and can be used as building blocks for developing an application.

You can create multiple snapshots of the same persistent volume claim (PVC). For CephFS, you can create up to 100 snapshots per PVC. For RADOS Block Device (RBD), you can create up to 512 snapshots per PVC.

> **NOTE**
>
> You cannot schedule periodic creation of snapshots.

11.1. CREATING VOLUME SNAPSHOTS

You can create a volume snapshot either from the Persistent Volume Claim (PVC) page or the Volume Snapshots page.

Prerequisites

- PVC must be in **Bound** state and must not be in use.

> **NOTE**
>
> OpenShift Container Storage only provides crash consistency for a volume snapshot of a PVC if a pod is using it. For application consistency, be sure to first tear down a running pod to ensure consistent snapshots or use any quiesce mechanism provided by the application to ensure it.

Procedure

From the Persistent Volume Claims page

1. Click **Storage → Persistent Volume Claims** from the OpenShift Web Console.
2. To create a volume snapshot, do one of the following:
   - Beside the desired PVC, click Action menu **(⋮) → Create Snapshot**.
   - Click on the PVC for which you want to create the snapshot and click **Actions → Create Snapshot**.
3. Enter a **Name** for the volume snapshot.
4. Choose the **Snapshot Class** from the drop-down list.
5. Click **Create**. You will be redirected to the Details page of the volume snapshot that is created.

From the Volume Snapshots page

1. Click **Storage → Volume Snapshots** from the OpenShift Web Console.
2. In the **Volume Snapshots** page, click **Create Volume Snapshot**

3. Choose the required **Project** from the drop-down list.

4. Choose the **Persistent Volume Claim** from the drop-down list.

5. Enter a **Name** for the snapshot.

6. Choose the **Snapshot Class** from the drop-down list.

7. Click **Create**. You will be redirected to the Details page of the volume snapshot that is created.

Verification steps

- Go to the **Details** page of the PVC and click the **Volume Snapshots** tab to see the list of volume snapshots. Verify that the new volume snapshot is listed.

- Click **Storage → Volume Snapshots** from the OpenShift Web Console. Verify that the new volume snapshot is listed.

- Wait for the volume snapshot to be in **Ready** state.

11.2. RESTORING VOLUME SNAPSHOTS

When you restore a volume snapshot, a new Persistent Volume Claim (PVC) gets created. The restored PVC is independent of the volume snapshot and the parent PVC.

You can restore a volume snapshot from either the Persistent Volume Claim page or the Volume Snapshots page.

Procedure

From the Persistent Volume Claims page

You can restore volume snapshot from the Persistent Volume Claims page only if the parent PVC is present.

1. Click **Storage → Persistent Volume Claims** from the OpenShift Web Console.

2. Click on the PVC name which has the volume snapshot that needs to be restored as a new PVC.

3. In the **Volume Snapshots** tab, beside the desired volume snapshot, click Action menu (⋮) → **Restore as new PVC**.

4. Enter a name for the new PVC.

5. Select the **Storage Class** name.

   NOTE

   (For Rados Block Device (RBD)) You must select a storage class with the same pool as that of the parent PVC.

6. Click **Restore**. You will be redirected to the new PVC details page.

From the Volume Snapshots page

1. Click **Storage → Volume Snapshots** from the OpenShift Web Console.
2. Beside the desired volume snapshot click Action Menu (⋮) → **Restore as new PVC**.
3. Enter a name for the new PVC.
4. Select the **Storage Class** name.

   NOTE

   (For Rados Block Device (RBD)) You must select a storage class with the same pool as that of the parent PVC.

5. Click **Restore**. You will be redirected to the new PVC details page.

NOTE

When you restore volume snapshots, the PVCs are created with the access mode of the parent PVC only if the parent PVC exists. Otherwise, the PVCs are created only with the ReadWriteOnce (RWO) access mode. Currently, you cannot specify the access mode using the OpenShift Web Console. However, you can specify the access mode from the CLI using the YAML. For more information, see Restoring a volume snapshot .

Verification steps

- Click **Storage → Persistent Volume Claims** from the OpenShift Web Console and confirm that the new PVC is listed in the **Persistent Volume Claims** page.
- Wait for the new PVC to reach **Bound** state.

11.3. DELETING VOLUME SNAPSHOTS

Prerequisites

- For deleting a volume snapshot, the volume snapshot class which is used in that particular volume snapshot should be present.

Procedure

From Persistent Volume Claims page

1. Click **Storage → Persistent Volume Claims** from the OpenShift Web Console.
2. Click on the PVC name which has the volume snapshot that needs to be deleted.
3. In the **Volume Snapshots** tab, beside the desired volume snapshot, click Action menu **(⋮)** → **Delete Volume Snapshot**

From Volume Snapshots page

1. Click **Storage → Volume Snapshots** from the OpenShift Web Console.

2. In the **Volume Snapshots** page, beside the desired volume snapshot click Action menu (⋮) → **Delete Volume Snapshot**

Verfication steps

- Ensure that the deleted volume snapshot is not present in the **Volume Snapshots** tab of the PVC details page.
- Click **Storage → Volume Snapshots** and ensure that the deleted volume snapshot is not listed.

CHAPTER 12. VOLUME CLONING

A clone is a duplicate of an existing storage volume that is used as any standard volume. You create a clone of a volume to make a point in time copy of the data. A persistent volume claim (PVC) cannot be cloned with a different size. You can create up to 512 clones per PVC for both CephFS and RADOS Block Device (RBD).

12.1. CREATING A CLONE

Prerequisites

- Source PVC must be in **Bound** state and must not be in use.

> **NOTE**
>
> Do not create a clone of a PVC if a Pod is using it. Doing so might cause data corruption because the PVC is not quiesced (paused).

Procedure

1. Click **Storage → Persistent Volume Claims** from the OpenShift Web Console.
2. To create a clone, do one of the following:
   - Beside the desired PVC, click Action menu **(⋮) → Clone PVC**.
   - Click on the PVC that you want to clone and click **Actions → Clone PVC**.
3. Enter a **Name** for the clone.
4. Click **Clone**. You will be redirected to the new PVC details page.

   > **NOTE**
   >
   > Clones are created with the access mode of the parent PVC. Currently, you cannot specify the access mode using the OpenShift Web Console UI. However, you can specify the access mode from the CLI using the YAML. For more information, see Provisioning a CSI volume clone .

5. Wait for the cloned PVC status to become **Bound**.
   The cloned PVC is now available to be consumed by the pods. This cloned PVC is independent of its dataSource PVC.

CHAPTER 13. REPLACING STORAGE NODES

- To replace an operational node, see Section 13.1, "Replacing an operational node on Red Hat Virtualization installer-provisioned infrastructure"
- To replace a failed node, see Section 13.2, "Replacing a failed node on Red Hat Virtualization installer-provisioned infrastructure"

13.1. REPLACING AN OPERATIONAL NODE ON RED HAT VIRTUALIZATION INSTALLER-PROVISIONED INFRASTRUCTURE

Use this procedure to replace an operational node on Red Hat Virtualization installer-provisioned infrastructure (IPI).

Prerequisites

- Red Hat recommends that replacement nodes are configured with similar infrastructure and resources to the node being replaced.
- You must be logged into the OpenShift Container Platform (RHOCP) cluster.

Procedure

1. Log in to OpenShift Web Console and click **Compute → Nodes**.
2. Identify the node that needs to be replaced. Take a note of its Machine Name.
3. Get labels on the node to be replaced.

```
$ oc get nodes --show-labels | grep <node_name>
```

4. Identify the mon (if any) and OSDs that are running in the node to be replaced.

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

5. Scale down the deployments of the pods identified in the previous step.
   For example:

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
--replicas=0 -n openshift-storage
```

6. Mark the nodes as unschedulable.

```
$ oc adm cordon <node_name>
```

7. Drain the node.

```
$ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets
```

8. Click **Compute → Machines**. Search for the required machine.

9. Besides the required machine, click the **Action menu (⋮) → Delete Machine**

10. Click **Delete** to confirm the machine deletion. A new machine is automatically created.

11. Wait for the new machine to start and transition into Running state.

IMPORTANT

This activity may take at least 5-10 minutes or more.

12. Click **Compute → Nodes** in the OpenShift web console. Confirm if the new node is in **Ready** state.

13. Apply the OpenShift Container Storage label to the new node using any one of the following:

From User interface

a. For the new node, click **Action Menu (⋮) → Edit Labels**

b. Add **cluster.ocs.openshift.io/openshift-storage** and click **Save**.

From Command line interface

- Execute the following command to apply the OpenShift Container Storage label to the new node:

```
$ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
```

14. Add the local storage devices available on these worker nodes to the OpenShift Container Storage StorageCluster.

a. Determine which **localVolumeSet** to edit.
Replace *local-storage-project* in the following commands with the name of your local storage project. The default project name is **openshift-local-storage** in OpenShift Container Storage 4.6 and later. Previous versions use **local-storage** by default.

```
# oc get -n local-storage-project localvolumeset
NAME         AGE
localblock   25h
```

b. Add the new node to the **localVolumeSet** definition.

```
# oc edit -n local-storage-project localvolumeset localblock
[...]
   nodeSelector:
    nodeSelectorTerms:
      - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
            - server1.example.com
            - server2.example.com
```

```
        #  - server3.example.com
           - newnode.example.com
[...]
```

Remember to save before exiting the editor.

15. Verify that the new **localblock** PV is available.

```
$ oc get pv | grep localblock
         CAPA- ACCESS RECLAIM                        STORAGE
NAME     CITY  MODES  POLICY  STATUS    CLAIM        CLASS      AGE
local-pv- 931Gi  RWO   Delete  Bound     openshift-storage/  localblock  25h
3e8964d3                          ocs-deviceset-2-0
                                  -79j94
local-pv- 931Gi  RWO   Delete  Bound     openshift-storage/  localblock  25h
414755e0                          ocs-deviceset-1-0
                                  -959rp
local-pv- 931Gi RWO Delete Available localblock 3m24s b481410
local-pv- 931Gi  RWO   Delete  Bound     openshift-storage/  localblock  25h
d9c5cbd6                          ocs-deviceset-0-0
                                  -nvs68
```

16. Change to the **openshift-storage** project.

```
$ oc project openshift-storage
```

17. Remove the failed OSD from the cluster.

```
$ oc process -n openshift-storage ocs-osd-removal \
-p FAILED_OSD_IDS=failed-osd-id1,failed-osd-id2 | oc create -f -
```

18. Verify that the OSD was removed successfully by checking the status of the **ocs-osd-removal** pod.
A status of **Completed** confirms that the OSD removal job succeeded.

```
# oc get pod -l job-name=ocs-osd-removal-failed-osd-id -n openshift-storage
```

NOTE

If **ocs-osd-removal** fails and the pod is not in the expected **Completed** state, check the pod logs for further debugging. For example:

```
# oc logs -l job-name=ocs-osd-removal-failed-osd_id -n openshift-storage --
tail=-1
```

19. Delete the PV associated with the failed node.

    a. Identify the PV associated with the PVC.

    ```
    # oc get -n openshift-storage pvc claim-name
    ```

    For example:

```
# oc get -n openshift-storage pvc ocs-deviceset-0-0-nvs68
                                    ACCESS STORAGE
NAME          STATUS    VOLUME    CAPACITY MODES  CLASS     AGE
ocs-deviceset- Released  local-pv- 931Gi    RWO    localblock 24h
0-0-nvs68               d9c5cbd6
```

b. Delete the PV.

```
# oc delete pv <persistent-volume>
```

For example:

```
# oc delete pv local-pv-d9c5cbd6
persistentvolume "local-pv-d9c5cbd6" deleted
```

20. Delete the **crashcollector** pod deployment.

```
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=failed-node-name -n openshift-storage
```

21. Deploy the new OSD by restarting the **rook-ceph-operator** to force operator reconciliation.

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

Example output:

```
NAME                               READY   STATUS    RESTARTS   AGE
rook-ceph-operator-6f74fb5bff-2d982   1/1     Running   0          1d20h
```

a. Delete the **rook-ceph-operator**.

```
# oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

Example output:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

b. Verify that the **rook-ceph-operator** pod is restarted.

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

Example output:

```
NAME                               READY   STATUS    RESTARTS   AGE
rook-ceph-operator-6f74fb5bff-7mvrq   1/1     Running   0          66s
```

Creation of the new OSD and **mon** might take several minutes after the operator restarts.

22. Delete the **ocs-osd-removal** job.

```
# oc delete job ocs-osd-removal-${osd_id_to_remove}
```

Example output:

```
job.batch "ocs-osd-removal-0" deleted
```

Verification steps

1. Execute the following command and verify that the new node is present in the output:

```
$ oc get nodes --show-labels | grep cluster.ocs.openshift.io/openshift-storage= |cut -d' ' -f1
```

2. Click **Workloads → Pods**, confirm that at least the following pods on the new node are in **Running** state:
   - **csi-cephfsplugin-***
   - **csi-rbdplugin-***
3. Verify that all other required OpenShift Container Storage pods are in Running state.
   Ensure that the new incremental **mon** is created and is in the Running state.

```
$ oc get pod -n openshift-storage | grep mon
```

Example output:

```
rook-ceph-mon-c-64556f7659-c2ngc                1/1     Running     0       6h14m
rook-ceph-mon-d-7c8b74dc4d-tt6hd                1/1     Running     0       4h24m
rook-ceph-mon-e-57fb8c657-wg5f2                 1/1     Running     0       162m
```

OSD and Mon might take several minutes to get to the **Running** state.

4. Verify that new OSD pods are running on the replacement node.

```
$ oc get pods -o wide -n openshift-storage| egrep -i new-node-name | egrep osd
```

5. (Optional) If data encryption is enabled on the cluster, verify that the new OSD devices are encrypted.
   For each of the new nodes identified in previous step, do the following:
   a. Create a debug pod and open a chroot environment for the selected host(s).

```
$ oc debug node/<node name>
$ chroot /host
```

   b. Run “lsblk” and check for the “crypt” keyword beside the **ocs-deviceset** name(s)

```
$ lsblk
```

6. If verification steps fail, contact Red Hat Support.

13.2. REPLACING A FAILED NODE ON RED HAT VIRTUALIZATION INSTALLER-PROVISIONED INFRASTRUCTURE

The ephemeral storage of Red Hat Virtualization for OpenShift Container Storage might cause data loss when there is an instance power off. Use this procedure to recover from such an instance power off on Red Hat Virtualization platform.

Prerequisites

- Red Hat recommends that replacement nodes are configured with similar infrastructure and resources to the node being replaced.
- You must be logged into the OpenShift Container Platform (RHOCP) cluster.

Procedure

1. Log in to OpenShift Web Console and click **Compute → Nodes**.
2. Identify the node that needs to be replaced. Take a note of its Machine Name.
3. Get the labels on the node to be replaced.

```
$ oc get nodes --show-labels | grep <node_name>
```

4. Identify the mon (if any) and OSDs that are running in the node to be replaced.

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

5. Scale down the deployments of the pods identified in the previous step.
   For example:

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
--replicas=0 -n openshift-storage
```

6. Mark the node as unschedulable.

```
$ oc adm cordon <node_name>
```

7. Remove the pods which are in Terminating state.

```
$ oc get pods -A -o wide | grep -i <node_name> |  awk '{if ($4 == "Terminating") system ("oc -
n " $1 " delete pods " $2  " --grace-period=0 " " --force ")}'
```

8. Drain the node.

```
$ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets
```

9. Click **Compute → Machines**. Search for the required machine.
10. Besides the required machine, click the **Action menu (⋮) → Delete Machine**
11. Click **Delete** to confirm the machine deletion. A new machine is automatically created.
12. Wait for the new machine to start and transition into Running state.

IMPORTANT

This activity may take at least 5-10 minutes or more.

13. Click **Compute → Nodes** in the OpenShift web console. Confirm if the new node is in **Ready** state.

14. Apply the OpenShift Container Storage label to the new node using any one of the following:

    From User interface

    a. For the new node, click **Action Menu (⋮) → Edit Labels**

    b. Add **cluster.ocs.openshift.io/openshift-storage** and click **Save**.

    From Command line interface

    - Execute the following command to apply the OpenShift Container Storage label to the new node:

    ```
    $ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
    ```

15. Add the local storage devices available in the new worker node to the OpenShift Container Storage StorageCluster.

16. Add the local storage devices available on these worker nodes to the OpenShift Container Storage StorageCluster.

    a. Determine which **localVolumeSet** to edit.
    Replace *local-storage-project* in the following commands with the name of your local storage project. The default project name is **openshift-local-storage** in OpenShift Container Storage 4.6 and later. Previous versions use **local-storage** by default.

    ```
    # oc get -n local-storage-project localvolumeset
    NAME         AGE
    localblock   25h
    ```

    b. Add the new node to the **localVolumeSet** definition.

    ```
    # oc edit -n local-storage-project localvolumeset localblock
    [...]
      nodeSelector:
       nodeSelectorTerms:
         - matchExpressions:
             - key: kubernetes.io/hostname
               operator: In
               values:
               - server1.example.com
               - server2.example.com
            #  - server3.example.com
               - newnode.example.com
    [...]
    ```

    Remember to save before exiting the editor.

17. Verify that the new **localblock** PV is available.

```
$ oc get pv | grep localblock
        CAPA- ACCESS RECLAIM                          STORAGE
NAME     CITY  MODES  POLICY  STATUS     CLAIM              CLASS      AGE
local-pv- 931Gi  RWO   Delete  Bound      openshift-storage/  localblock  25h
3e8964d3                              ocs-deviceset-2-0
                               -79j94
local-pv- 931Gi  RWO   Delete  Bound      openshift-storage/  localblock  25h
414755e0                              ocs-deviceset-1-0
                               -959rp
local-pv- 931Gi RWO Delete Available localblock 3m24s b481410
local-pv- 931Gi  RWO   Delete  Bound      openshift-storage/  localblock  25h
d9c5cbd6                              ocs-deviceset-0-0
                               -nvs68
```

18. Change to the **openshift-storage** project.

```
$ oc project openshift-storage
```

19. Remove the failed OSD from the cluster.

```
$ oc process -n openshift-storage ocs-osd-removal \
-p FAILED_OSD_IDS=failed-osd-id1,failed-osd-id2 | oc create -f -
```

20. Verify that the OSD was removed successfully by checking the status of the **ocs-osd-removal** pod.
A status of **Completed** confirms that the OSD removal job succeeded.

```
# oc get pod -l job-name=ocs-osd-removal-failed-osd-id -n openshift-storage
```

NOTE

If **ocs-osd-removal** fails and the pod is not in the expected **Completed** state, check the pod logs for further debugging. For example:

```
# oc logs -l job-name=ocs-osd-removal-failed-osd_id -n openshift-storage --
tail=-1
```

21. Delete the PV associated with the failed node.

    a. Identify the PV associated with the PVC.

    ```
    # oc get -n openshift-storage pvc claim-name
    ```

    For example:

    ```
    # oc get -n openshift-storage pvc ocs-deviceset-0-0-nvs68
                                        ACCESS STORAGE
    NAME          STATUS    VOLUME    CAPACITY MODES  CLASS      AGE
    ocs-deviceset- Released  local-pv- 931Gi    RWO    localblock 24h
    0-0-nvs68                d9c5cbd6
    ```

b. Delete the PV.

```
# oc delete pv <persistent-volume>
```

For example:

```
# oc delete pv local-pv-d9c5cbd6
persistentvolume "local-pv-d9c5cbd6" deleted
```

22. Delete the **crashcollector** pod deployment.

```
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=failed-node-name -n openshift-storage
```

23. Deploy the new OSD by restarting the **rook-ceph-operator** to force operator reconciliation.

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

Example output:

```
NAME                                  READY   STATUS    RESTARTS   AGE
rook-ceph-operator-6f74fb5bff-2d982   1/1     Running   0          1d20h
```

a. Delete the **rook-ceph-operator**.

```
# oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

Example output:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

b. Verify that the **rook-ceph-operator** pod is restarted.

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

Example output:

```
NAME                                  READY   STATUS    RESTARTS   AGE
rook-ceph-operator-6f74fb5bff-7mvrq   1/1     Running   0          66s
```

Creation of the new OSD and **mon** might take several minutes after the operator restarts.

24. Delete the`ocs-osd-removal` job.

```
# oc delete job ocs-osd-removal-${osd_id_to_remove}
```

Example output:

```
job.batch "ocs-osd-removal-0" deleted
```

Verification steps

1. Execute the following command and verify that the new node is present in the output:

```
$ oc get nodes --show-labels | grep cluster.ocs.openshift.io/openshift-storage= |cut -d' ' -f1
```

2. Click **Workloads → Pods**, confirm that at least the following pods on the new node are in **Running** state:

   - **csi-cephfsplugin-***
   - **csi-rbdplugin-***

3. Verify that all other required OpenShift Container Storage pods are in Running state.
   Ensure that the new incremental **mon** is created and is in the Running state.

```
$ oc get pod -n openshift-storage | grep mon
```

   Example output:

```
rook-ceph-mon-c-64556f7659-c2ngc                    1/1     Running     0       6h14m
rook-ceph-mon-d-7c8b74dc4d-tt6hd                   1/1     Running     0       4h24m
rook-ceph-mon-e-57fb8c657-wg5f2                    1/1     Running     0       162m
```

   OSD and Mon might take several minutes to get to the **Running** state.

4. Verify that new OSD pods are running on the replacement node.

```
$ oc get pods -o wide -n openshift-storage| egrep -i new-node-name | egrep osd
```

5. (Optional) If data encryption is enabled on the cluster, verify that the new OSD devices are encrypted.
   For each of the new nodes identified in previous step, do the following:

   a. Create a debug pod and open a chroot environment for the selected host(s).

```
$ oc debug node/<node name>
$ chroot /host
```

   b. Run “lsblk” and check for the “crypt” keyword beside the **ocs-deviceset** name(s)

```
$ lsblk
```

6. If verification steps fail, contact Red Hat Support.

CHAPTER 14. REPLACING FAILED STORAGE DEVICES ON RED HAT VIRTUALIZATION PLATFORM

When you need to replace a storage device on Red Hat Virtualization platform, you must replace the storage node. For information about how to replace nodes, see Replacing failed storage nodes on Red Hat Virtualization platform.

CHAPTER 15. UPDATING OPENSHIFT CONTAINER STORAGE

15.1. OVERVIEW OF THE OPENSHIFT CONTAINER STORAGE UPDATE PROCESS

You can upgrade Red Hat OpenShift Container Storage and its components, either between minor releases like 4.5 and 4.6, or between batch updates like 4.6.0 and 4.6.1.

You need to upgrade the different parts of OpenShift Container Storage in a specific order.

1. **Update OpenShift Container Platform** according to the Updating clusters documentation for OpenShift Container Platform.
2. **Update OpenShift Container Storage.**
   a. **Update the OpenShift Container Storage operator**, using the appropriate process for your setup:
      - To prepare a disconnected or proxy environment for updates, see Operators guide to using Operator Lifecycle Manager on restricted networks .
      - Update OpenShift Container Storage in internal mode

   b. **If you use local storage:**
      i. **Update the Local Storage operator**
      See Checking for Local Storage Operator deployments if you are unsure.
      ii. **Perform post-update configuration changes** for clusters backed by local storage.
      See Post-update configuration for clusters backed by local storage for details.

Update considerations

Review the following important considerations before you begin.

- Red Hat recommends using the same version of Red Hat OpenShift Container Platform with Red Hat OpenShift Container Storage.
  See the Interoperability Matrix for more information about supported combinations of OpenShift Container Platform and OpenShift Container Storage.
- The Local Storage Operator is fully supported only when the Local Storage Operator version matches the Red Hat OpenShift Container Platform version.

15.2. PREPARING TO UPDATE IN A DISCONNECTED ENVIRONMENT

When your Red Hat OpenShift Container Storage environment is not directly connected to the internet, some additional configuration is required to provide the Operator Lifecycle Manager (OLM) with alternatives to the default Operator Hub and image registries.

See the OpenShift Container Platform documentation for more general information: Updating an Operator catalog image.

To configure your cluster for disconnected update:

1. Configure authentication for an alternative registry.

2. Build and mirror the Red Hat operator catalog .
3. Creating Operator imageContentSourcePolicy
4. Updating redhat-operator catalogsource

When these steps are complete, Continue with update as usual.

15.2.1. Adding mirror registry authentication details

Prerequisites

- Verify that your existing disconnected cluster uses OpenShift Container Platform 4.3 or higher.
- Verify that you have an **oc client** version of 4.4 or higher.
- Prepare a mirror host with a mirror registry. See Preparing your mirror host for details.

Procedure

1. Log in to the OpenShift Container Platform cluster using the **cluster-admin** role.
2. Locate your **auth.json** file.
   This file is generated when you use podman or docker to log in to a registry. It is located in one of the following locations:
   - **~/.docker/auth.json**
   - **/run/user/<UID>/containers/auth.json**
   - **/var/run/containers/<UID>/auth.json**
3. Obtain your unique Red Hat registry pull secret and paste it into your **auth.json** file. It will look something like this.

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "*****************",
      "email": "user@example.com"
    },
    "quay.io": {
      "auth": "*****************",
      "email": "user@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "*****************",
      "email": "user@example.com"
    },
    "registry.redhat.io": {
      "auth": "*****************",
      "email": "user@example.com"
    }
  }
}
```

4. Export environment variables with the appropriate details for your setup.

```
$ export AUTH_FILE="<location_of_auth.json>"
$ export MIRROR_REGISTRY_DNS="<your_registry_url>:<port>"
```

5. Use **podman** to log in to the mirror registry and store the credentials in the **${AUTH_FILE}**.

```
$ podman login ${MIRROR_REGISTRY_DNS} --tls-verify=false --authfile ${AUTH_FILE}
```

This adds the mirror registry to the **auth.json** file.

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "*****************",
      "email": "user@example.com"
    },
    "quay.io": {
      "auth": "*****************",
      "email": "user@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "*****************",
      "email": "user@example.com"
    },
    "registry.redhat.io": {
      "auth": "*****************",
      "email": "user@example.com"
    },
    "<mirror_registry>": {
      "auth": "*****************",
    }
  }
}
```

15.2.2. Building and mirroring the Red Hat operator catalog

Follow this process on a host that has access to Red Hat registries to create a mirror of those registries.

Prerequisites

- Run these commands as a cluster administrator.
- Be aware that mirroring the **redhat-operator** catalog can take hours to complete, and requires substantial available disk space on the mirror host.

Procedure

1. Build the catalog for **redhat-operators**.
   Set **--from** to the **ose-operator-registry** base image using the tag that matches the target OpenShift Container Platform cluster major and minor version.

```
$ oc adm catalog build --appregistry-org redhat-operators \
  --from=registry.redhat.io/openshift4/ose-operator-registry:v4.6 \
```

```
--to=${MIRROR_REGISTRY_DNS}/olm/redhat-operators:v2 \
--registry-config=${AUTH_FILE} \
--filter-by-os="linux/amd64" --insecure
```

2. Mirror the catalog for **redhat-operators**.
   This is a long operation and can take 1-5 hours. Make sure there is 100 GB available disk space on the mirror host.

```
$ oc adm catalog mirror ${MIRROR_REGISTRY_DNS}/olm/redhat-operators:v2 \
${MIRROR_REGISTRY_DNS} --registry-config=${AUTH_FILE} --insecure
```

15.2.3. Creating Operator imageContentSourcePolicy

After the **oc adm catalog mirror** command is completed, the **imageContentSourcePolicy.yaml** file gets created. The output directory for this file is usually, **./[catalog image name]-manifests)**. Use this procedure to add any missing entries to the **.yaml** file and apply them to cluster.

Procedure

1. Check the content of this file for the mirrors mapping shown as follows:

```
spec:
  repositoryDigestMirrors:
    - mirrors:
      - <your_registry>/ocs4
      source: registry.redhat.io/ocs4
    - mirrors:
      - <your_registry>/rhceph
      source: registry.redhat.io/rhceph
    - mirrors:
      - <your_registry>/openshift4
      source: registry.redhat.io/openshift4
    - mirrors:
      - <your_registry>/rhscl
      source: registry.redhat.io/rhscl
```

2. Add any missing entries to the end of the **imageContentSourcePolicy.yaml** file.

3. Apply the imageContentSourcePolicy.yaml file to the cluster.

```
$ oc apply -f ./[output dir]/imageContentSourcePolicy.yaml
```

   Once the Image Content Source Policy is updated, all the nodes (master, infra, and workers) in the cluster need to be updated and rebooted. This process is automatically handled through the Machine Config Pool operator and take up to 30 minutes although the exact elapsed time might vary based on the number of nodes in your OpenShift cluster. You can monitor the update process by using the **oc get mcp** command or the **oc get node** command.

15.2.4. Updating redhat-operator CatalogSource

Procedure

1. Recreate a **CatalogSource** object that references the catalog image for Red Hat operators.

> **NOTE**
>
> Make sure you have mirrored the correct catalog source with the correct version (that is, **v2**).

Save the following in a **redhat-operator-catalogsource.yaml** file, remembering to replace *<your_registry>* with your mirror registry URL:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: redhat-operators
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  icon:
    base64data:
PHN2ZyBpZD0iTGF5ZXJfMSIgZGF0YS1uYW1lPSJMYXllciAxIiB4bWxucz0iaHR0cDovL3d3dy
53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMCAxOTIgMTQ1Ij48ZGVmcz48c3R5bGU+L
mNscy0xe2ZpbGw6I2UwMDt9PC9zdHlsZT48L2RlZnM+PHRpdGxlPlJlZEhhdC1Mb2dvLUhhd
C1Db2xvcjwvdGl0bGU+PHBhdGggZD0iTTE1Ny43Nyw2Mi42MWExNCwxNCwwLDAsMSwuM
zEsMy40MmMwLDE0Ljg4LTE4LjEsMTcuNDYtMzAuNjEsMTcuNDZDNzguODMsODMuNDksN
DIuNTMsNTMuMjYsNDIuNTMsNDRhNi40Myw2LjQzLDAsMCwxLC4yMi0xLjk0bC0zLjY2LDku
MDZhMTguNDUsMTguNDUsMCwwLDAtMS41MSw3LjMzYzAsMTguMTEsNDEsNDUuNDgs
ODcuNzQsNDUuNDgsMjAuNjksMCwzNi40My03Ljc2LDM2LjQzLTIxLjc3LDAtMS4wOCwwLTE
uOTQtMS43My0xMC4xM1oiLz48cGF0aCBjbGFzcz0iY2xzLTEiIGQ9Ik0xMjcuNDcsODMuNDlj
MTIuNTEsMCwzMC42MS0yLjU4LDMwLjYxLTE3LjQ2YTE0LDE0LDAsMCwwLS4zMS0zLjQyb
C03LjQ1LTMyLjM2Yy0xLjcyLTcuMTItMy4yMy0xMC4zNS0xNS43My0xNi42QzEyNC44OSw4Lj
Y5LDEwMy43Ni41LDk3LjUxLjUsOTEuNjkuNSw5MCw4LDgzLjA2LDhjLTYuNjgsMC0xMS42N
C01LjYtMTcuODktNS42LTYsMC05LjkxLDQuMDktMTIuOTMsMTIuNSwwLDAtOC40MSwyMy
43Mi05LjQ5LDI3LjE2QTYuNDMsNi40MywwLDAsMCw0Mi41Myw0NGMwLDkuMjIsMzYuMywz
OS40NSw4NC45NCwzOS40NU0xNjAsNzIuMDdjMS43Myw4LjE5LDEuNzMsOS4wNSwxLjczL
DEwLjEzLDAsMTQtMTUuNzQsMjEuNzctMzYuNDMsMjEuNzdDNzguNTQsMTA0LDM3LjU4L
Dc2LjYsMzcuNTgsNTguNDlhMTguNDUsMTguNDUsMCwwLDEsMS41MS03LjMzQzIyLjI3LDU
yLC41LDU1LC41LDc0LjIyYzAsMzEuNDgsNzQuNTksNzAuMjgsMTMzLjY1LDcwLjI4LDQ1LjI4L
DAsNTYuNy0yMC40OCw1Ni43LTM2LjY1LDAtMTIuNzItMTEtMjcuMTYtMzAuODMtMzUuNzgi
Lz48L3N2Zz4=
    mediatype: image/svg+xml
  image: <your_registry>/olm/redhat-operators:v2
  displayName: Redhat Operators Catalog
  publisher: Red Hat
```

2. Create a catalogsource using the redhat-operator-catalogsource.yaml file:

```
$ oc apply -f redhat-operator-catalogsource.yaml
```

3. Verify that the new **redhat-operator** pod is running.

```
$ oc get pod -n openshift-marketplace | grep redhat-operators
```

15.2.5. Continue to update

After your alternative catalog source is configured, you can continue to the appropriate update process:

- Updating OpenShift Container Storage in internal mode

15.3. UPDATING OPENSHIFT CONTAINER STORAGE IN INTERNAL MODE

Use the following procedures to update your OpenShift Container Storage cluster deployed in internal mode.

15.3.1. Enabling automatic updates for OpenShift Container Storage operator in internal mode

Use this procedure to enable automatic update approval for updating OpenShift Container Storage operator in OpenShift Container Platform.

Prerequisites

- Under **Persistent Storage** in the **Status** card, confirm that the *OCS Cluster* and *Data Resiliency* has a green tick mark.
- Under **Object Service** in the **Status** card, confirm that both the *Object Service* and *Data Resiliency* are in **Ready** state (green tick).
- Update the OpenShift Container Platform cluster to the latest stable release of version 4.5.X or 4.6.Y, see Updating Clusters.
- Switch the Red Hat OpenShift Container Storage channel from **stable-4.5** to **stable-4.6**. For details about channels, see OpenShift Container Storage upgrade channels and releases .

> **NOTE**
>
> You are required to switch channels only when you are updating minor versions (for example, updating from 4.5 to 4.6) and not when updating between batch updates of 4.6 (for example, updating from 4.6.0 to 4.6.1).

- Ensure that all OpenShift Container Storage Pods, including the operator pods, are in **Running** state in the **openshift-storage namespace**.
  To view the state of the pods, click **Workloads → Pods** from the left pane of the OpenShift Web Console. Select **openshift-storage** from the **Project** drop down list.
- Ensure that you have sufficient time to complete the Openshift Container Storage update process, as the update time varies depending on the number of OSDs that run in the cluster.

Procedure

1. Log in to OpenShift Web Console.
2. Click **Operators → Installed Operators**
3. Select the **openshift-storage** project.
4. Click the OpenShift Container Storage operator name.
5. Click the **Subscription** tab and click the link under **Approval**.

6. Select **Automatic (default)** and click **Save**.

7. Perform one of the following depending on the **Upgrade Status**:

   - **Upgrade Status** *shows* **requires approval**.

     NOTE

     Upgrade status shows **requires approval** if the new OpenShift Container Storage version is already detected in the channel, and approval strategy was changed from **Manual** to **Automatic** at the time of update.

     a. Click on the **Install Plan** link.

     b. On the **InstallPlan Details** page, click **Preview Install Plan**.

     c. Review the install plan and click **Approve**.

     d. Wait for the **Status** to change from **Unknown** to **Created**.

     e. Click **Operators → Installed Operators**

     f. Select the **openshift-storage** project.

     g. Wait for the **Status** to change to **Up to date**

   - **Upgrade Status** *does not show* **requires approval**:

     a. Wait for the update to initiate. This may take up to 20 minutes.

     b. Click **Operators → Installed Operators**

     c. Select the **openshift-storage** project.

     d. Wait for the **Status** to change to **Up to date**

Verification steps

1. Click **Overview → Persistent Storage** tab and in the **Status** card confirm that the *OCS Cluster* and *Data Resiliency* has a green tick mark indicating it is healthy.

2. Click **Overview → Object Service** tab and in the **Status** card confirm that both the *Object Service* and *Data Resiliency* are in **Ready** state (green tick) indicating it is healthy.

3. Click **Operators → Installed Operators → OpenShift Container Storage Operator**. Under **Storage Cluster**, verify that the cluster service status is **Ready**.

   NOTE

   Once updated from OpenShift Container Storage version 4.5 to 4.6, the **Version** field here will still display 4.5. This is because the **ocs-operator** does not update the string represented in this field.

4. Ensure that all OpenShift Container Storage Pods, including the operator pods, are in **Running** state in the **openshift-storage namespace**.

To view the state of the pods, click **Workloads → Pods**. Select **openshift-storage** from the **Project** drop down list.

5. If verification steps fail, contact Red Hat Support.

Additional Resources

If you face any issues while updating OpenShift Container Storage, see the *Commonly required logs for troubleshooting* section in the Troubleshooting guide.

15.3.2. Manually updating OpenShift Container Storage operator in internal mode

Use this procedure to update OpenShift Container Storage operator by providing manual approval to the install plan.

Prerequisites

- Under **Persistent Storage** in the **Status** card, confirm that the *OCS Cluster* and *Data Resiliency* has a green tick mark.
- Under **Object Service** in the **Status** card, confirm that both the *Object Service* and *Data Resiliency* are in **Ready** state (green tick).
- Update the OpenShift Container Platform cluster to the latest stable release of version 4.5.X or 4.6.Y, see Updating Clusters.
- Switch the Red Hat OpenShift Container Storage channel from **stable-4.5** to **stable-4.6**. For details about channels, see OpenShift Container Storage upgrade channels and releases.

> **NOTE**
>
> You are required to switch channels only when you are updating minor versions (for example, updating from 4.5 to 4.6) and not when updating between batch updates of 4.6 (for example, updating from 4.6.0 to 4.6.1).

- Ensure that all OpenShift Container Storage Pods, including the operator pods, are in **Running** state in the **openshift-storage namespace**.
  To view the state of the pods, click **Workloads → Pods** from the left pane of the OpenShift Web Console. Select **openshift-storage** from the **Project** drop down list.
- Ensure that you have sufficient time to complete the Openshift Container Storage update process, as the update time varies depending on the number of OSDs that run in the cluster.

Procedure

1. Log in to OpenShift Web Console.
2. Click **Operators → Installed Operators**
3. Select the **openshift-storage** project.
4. Click the **OpenShift Container Storage** operator name.
5. Click the **Subscription** tab and click the link under **Approval**.
6. Select **Manual** and click **Save**.

7. Wait for the **Upgrade Status** to change to **Upgrading**.

8. If the **Upgrade Status** shows **requires approval**, click on **requires approval**.

9. On the **InstallPlan Details** page, click **Preview Install Plan**.

10. Review the install plan and click **Approve**.

11. Wait for the **Status** to change from **Unknown** to **Created**.

12. Click **Operators → Installed Operators**

13. Select the **openshift-storage** project.

14. Wait for the **Status** to change to **Up to date**

Verification steps

1. Click **Overview → Persistent Storage** tab and in the **Status** card confirm that the *OCS Cluster* and *Data Resiliency* has a green tick mark indicating it is healthy.

2. Click **Overview → Object Service** tab and in the **Status** card confirm that both the *Object Service* and *Data Resiliency* are in **Ready** state (green tick) indicating it is healthy.

3. Click **Operators → Installed Operators → OpenShift Container Storage Operator**. Under **Storage Cluster**, verify that the cluster service status is **Ready**.

   NOTE

   Once updated from OpenShift Container Storage version 4.5 to 4.6, the **Version** field here will still display 4.5. This is because the **ocs-operator** does not update the string represented in this field.

4. Ensure that all OpenShift Container Storage Pods, including the operator pods, are in **Running** state in the **openshift-storage namespace**.
   To view the state of the pods, click **Workloads → Pods** from the left pane of the OpenShift Web Console. Select **openshift-storage** from the **Project** drop down list.

5. If verification steps fail, contact Red Hat Support.

Additional Resources

If you face any issues while updating OpenShift Container Storage, see the *Commonly required logs for troubleshooting* section in the Troubleshooting guide.

15.4. POST-UPDATE CONFIGURATION CHANGES

In some cases, additional configuration steps are required after an update to ensure that all features work as expected.

15.4.1. Post-update configuration for clusters backed by local storage

In Red Hat OpenShift Container Platform 4.6 and onward, the Local Storage operator provides new custom resource types for managing local storage:

- **LocalVolumeDiscovery**
- **LocalVolumeSet**

These resource types are not automatically handled as part of an update from earlier versions, and must be created manually.

15.4.1.1. Creating a LocalVolumeDiscovery custom resource using the command line

Create a **LocalVolumeDiscovery** custom resource to ensure that the device management user interface can discover the state of local devices and provide information about devices that are available on cluster nodes.

Prerequisites

- Administrative access to the OpenShift Container Platform cluster.

Procedure

1. Change into the project that has Local Storage operator installed.

   ```
   $ oc project local-storage-project
   ```

   Replace *local-storage-project* with the name of your Local Storage project.

   In version 4.5 and earlier the name of the default local storage project is **local-storage**. In version 4.6 and later, the name of the default local storage project is **openshift-local-storage**.

2. Define the **LocalVolumeDiscovery** custom resource.
   For example, define the following in a **local-volume-discovery.yaml** file.

   ```
   apiVersion: local.storage.openshift.io/v1alpha1
   kind: LocalVolumeDiscovery
   metadata:
     name: auto-discover-devices
   spec:
     nodeSelector:
       nodeSelectorTerms:
         - matchExpressions:
             - key: kubernetes.io/hostname
               operator: In
               values:
                 - worker1.example.com
                 - worker2.example.com
                 - worker3.example.com
   ```

3. Create the **LocalVolumeDiscovery** custom resource.

   ```
   $ oc create -f local-volume-discovery.yaml
   ```

Verification steps

1. Log in to the OpenShift web console.

2. Click **Compute → Node** and click on the name of the node.

3. Click the **Disks** tab and check that you can see the devices available on that node.

15.4.1.2. Creating a LocalVolumeSet custom resource using the command line

Create a **LocalVolumeSet** custom resource to automatically provision certain storage devices as persistent volumes based on criteria that you specify. Persistent volumes are created for any devices that match the **deviceInclusionSpec** criteria on any node that matches the **nodeSelector** criteria.

Prerequisites

- Administrative access to the OpenShift Container Platform cluster.

Procedure

1. Define a **LocalVolumeSet** custom resource in a **local-volume-set.yaml** file.

```
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeSet
metadata:
  name: localblock
spec:
  nodeSelector:
    nodeSelectorTerms:
      - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
              - worker1.example.com
              - worker2.example.com
              - worker3.example.com
  storageClassName: localblock
  volumeMode: Block
  maxDeviceCount: 10 # optional, limit devices provisioned per node
  deviceInclusionSpec:
    deviceTypes: # list of types to allow
      - disk
      - part # omit this to use only whole devices
    deviceMechanicalProperty:
      - NonRotational
    minSize: 100Gi # optional, minimum size of device to allow
    maxSize: 100Ti # optional, maximum size of device to allow
    models: # (optional) list of models to allow
      - SAMSUNG
      - Crucial_CT525MX3
    vendors: # (optional) list of device vendors to allow
      - ATA
      - ST2000LM
```

The above definition selects whole disks or partitions on specific models of non-rotational devices that are between 100 GB and 100 TB in size, provided by specific vendors, from the **worker1**, **worker2** and **worker3** nodes. The **localblock** storage class is created and persistent volumes are provisioned from discovered devices.

IMPORTANT

Select an appropriate value for **minSize** to ensure system partitions are not selected.

2. Create the **LocalVolumeSet**.

```
$ oc create -f local-volume-set.yaml
```

Verification steps

1. Use the following command to track provisioning of persistent volumes for devices that match the **deviceInclusionSpec**. It can take a few minutes to provision persistent volumes.

```
$ oc describe localvolumeset localblock
[...]
Status:
  Conditions:
    Last Transition Time:        2020-11-17T05:03:32Z
    Message:                     DiskMaker: Available, LocalProvisioner: Available
    Status:                      True
    Type:                        DaemonSetsAvailable
    Last Transition Time:        2020-11-17T05:03:34Z
    Message:                     Operator reconciled successfully.
    Status:                      True
    Type:                        Available
  Observed Generation:           1
  Total Provisioned Device Count: 4
Events:
Type    Reason      Age        From              Message
----    ------      ----       ----              -------
Normal  Discovered  2m30s (x4    localvolumeset-     ip-10-0-147-124.us-east-
        NewDevice   over 2m30s)  symlink-controller  2.compute.internal -
                                               found possible matching
                                               disk, waiting 1m to claim
Normal  FoundMatch  89s (x4      localvolumeset-     ip-10-0-147-124.us-east-
        ingDisk     over 89s)    symlink-controller  2.compute.internal -
                                               symlinking matching disk
```

2. Verify the state of the provisioned persistent volumes.

```
$ oc get pv
                    ACCESS   RECLAIM          STORAGE
NAME      CAPACITY  MODES    POLICY   STATUS     CLASS      AGE
local-pv-  500Gi    RWO      Delete   Available  localblock  7m48s
3584969f
local-pv-  500Gi    RWO      Delete   Available  localblock  7m48s
3aee84fa
local-pv-  500Gi    RWO      Delete   Available  localblock  7m48s
644d09ac
local-pv-  500Gi    RWO      Delete   Available  localblock  7m48s
c73cee1
```

15.4.1.3. Adding annotations

Use this procedure to add annotations to storage cluster to enable replacing of failed storage devices through the user interface when you upgraded to OpenShift Container Storage 4.6 from a previous version.

Procedure

1. Log in to OpenShift Container Platform Web Console.
2. Click **Home → Search**.
3. Search for **StorageCluster** in **Resources** and click on it.
4. Beside **ocs-storagecluster**, click Action menu **(⋮) → Edit annotations**.
5. Add **cluster.ocs.openshift.io/local-devices** and **true** for **KEY** and **VALUE** respectively.
6. Click **Save**.