Red Hat OpenShift Container Storage 4.4

Deploying and managing OpenShift Container Storage on Microsoft Azure

How to install and manage

Last Updated: 2021-03-12

Red Hat OpenShift Container Storage 4.4 Deploying and managing OpenShift Container Storage on Microsoft Azure

How to install and manage

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Read this document for instructions on installing and managing Red Hat OpenShift Container Storage 4.4 on Microsoft Azure. Deploying and managing OpenShift Container Storage on Microsoft Azure 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 ... 4

CHAPTER 1. PLANNING OPENSHIFT CONTAINER STORAGE DEPLOYMENT ON MICROSOFT AZURE ... 5
1.1. REQUIREMENTS FOR INSTALLING OPENSHIFT CONTAINER STORAGE ON MICROSOFT AZURE 5
1.2. SIZING AND SCALING 5
1.3. SUPPORTED WORKLOAD TYPES 6

CHAPTER 2. DEPLOYING OPENSHIFT CONTAINER STORAGE ON MICROSOFT AZURE ... 7
2.1. INSTALLING RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR USING THE OPERATOR HUB 7
2.2. CREATING AN OPENSHIFT CONTAINER STORAGE SERVICE 9
2.3. CREATING A NEW BACKING STORE 10
2.4. VERIFYING OPENSHIFT CONTAINER STORAGE DEPLOYMENT 13
2.4.1. Verifying the state of the pods 13
2.4.2. Verifying the OpenShift Container Storage cluster is healthy 15
2.4.3. Verifying the Multicloud Object Gateway is healthy 16
2.4.4. Verifying that the storage classes are created and listed 17

CHAPTER 3. UNINSTALLING OPENSHIFT CONTAINER STORAGE ... 18
3.1. REMOVING MONITORING STACK FROM OPENSHIFT CONTAINER STORAGE 24
3.2. REMOVING OPENSHIFT CONTAINER PLATFORM REGISTRY FROM OPENSHIFT CONTAINER STORAGE 28
3.3. REMOVING THE CLUSTER LOGGING OPERATOR FROM OPENSHIFT CONTAINER STORAGE 29

CHAPTER 4. CONFIGURE STORAGE FOR OPENSHIFT CONTAINER PLATFORM SERVICES ... 30
4.1. CONFIGURING IMAGE REGISTRY TO USE OPENSHIFT CONTAINER STORAGE 30
4.2. CONFIGURING MONITORING TO USE OPENSHIFT CONTAINER STORAGE 32
4.3. CLUSTER LOGGING FOR OPENSHIFT CONTAINER STORAGE 34
4.3.1. Configuring persistent storage 35
4.3.2. Configuring cluster logging to use OpenShift Container Storage 36

CHAPTER 5. BACKING OPENSHIFT CONTAINER PLATFORM APPLICATIONS WITH OPENSHIFT CONTAINER STORAGE ... 39

CHAPTER 6. SCALING STORAGE NODES ... 41
6.1. SCALING UP STORAGE BY ADDING CAPACITY TO YOUR OPENSHIFT CONTAINER STORAGE NODES ON AZURE INFRASTRUCTURE 41
6.2. SCALING OUT STORAGE CAPACITY BY ADDING NEW NODES 43
6.2.1. Adding a node on Azure installer-provisioned infrastructure 43
6.2.2. Verifying the addition of a new node 44
6.2.3. Scaling up storage capacity 44

CHAPTER 7. MULTICLOUD OBJECT GATEWAY ... 45
7.1. ABOUT THE MULTICLOUD OBJECT GATEWAY 45
7.2. ACCESSING THE MULTICLOUD OBJECT GATEWAY WITH YOUR APPLICATIONS 45
7.2.1. Accessing the Multicloud Object Gateway from the terminal 45
7.2.2. Accessing the Multicloud Object Gateway from the MCG command-line interface 47
7.3. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD 50
7.3.1. Adding storage resources for hybrid or Multicloud using the MCG command line interface 50
7.3.2. Creating an s3 compatible Multicloud Object Gateway backingstore 51
7.3.3. Adding storage resources for hybrid and Multicloud using the user interface 53
7.3.4. Creating a new bucket class 55
7.3.5. Creating a new backing store 57
7.4. MIRRORING DATA FOR HYBRID AND MULTICLOUD BUCKETS 59
7.4.1. Creating bucket classes to mirror data using the MCG command-line-interface 59

7.4.2. Creating bucket classes to mirror data using a YAML 60
7.4.3. Configuring buckets to mirror data using the user interface 60
7.5. BUCKET POLICIES IN THE MULTICLOUD OBJECT GATEWAY 62
7.5.1. About bucket policies 62
7.5.2. Using bucket policies 62
7.5.3. Creating an AWS S3 user in the Multicloud Object Gateway 64
7.6. OBJECT BUCKET CLAIM 66
7.6.1. Dynamic Object Bucket Claim 66
7.6.2. Creating an Object Bucket Claim using the command line interface 68
7.6.3. Creating an Object Bucket Claim using the OpenShift Web Console 71
7.6.3.1. Delete an Object Bucket Claim 73
7.6.3.2. Viewing object buckets using the Multicloud Object Gateway user interface 74
7.7. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS 75
7.7.1. S3 endpoints in the Multicloud Object Gateway 75
7.7.2. Scaling with storage nodes 75

CHAPTER 8. MANAGING PERSISTENT VOLUME CLAIMS 79
8.1. CONFIGURING APPLICATION PODS TO USE OPENSHIFT CONTAINER STORAGE 79
8.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS 80
8.3. REVIEWING PERSISTENT VOLUME CLAIM REQUEST EVENTS 81
8.4. DYNAMIC PROVISIONING 81
8.4.1. About dynamic provisioning 82
8.4.2. Dynamic provisioning in OpenShift Container Storage 82
8.4.3. Available dynamic provisioning plug-ins 82

CHAPTER 9. REPLACING STORAGE NODES 85
9.1. REPLACING OPERATIONAL NODES ON AZURE INSTALLER-PROVISIONED INFRASTRUCTURE 85
9.2. REPLACING FAILED NODES ON AZURE INSTALLER-PROVISIONED INFRASTRUCTURE 86

CHAPTER 10. REPLACING STORAGE DEVICES 88
10.1. REPLACING OPERATIONAL OR FAILED STORAGE DEVICES ON AZURE INSTALLER-PROVISIONED INFRASTRUCTURE 88

CHAPTER 11. UPDATING OPENSHIFT CONTAINER STORAGE 89
11.1. ENABLING AUTOMATIC UPDATES FOR OPENSHIFT CONTAINER STORAGE OPERATOR 89
11.2. MANUALLY UPDATING OPENSHIFT CONTAINER STORAGE OPERATOR 91

PREFACE

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

Red Hat 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

CHAPTER 1. PLANNING OPENSHIFT CONTAINER STORAGE DEPLOYMENT ON MICROSOFT AZURE

Use this section to understand the requirements to install OpenShift Container Storage on Microsoft Azure.

1.1. REQUIREMENTS FOR INSTALLING OPENSHIFT CONTAINER STORAGE ON MICROSOFT AZURE

Instance type	Standard_D16s_v3
Node	• CPU: 16 vCPUs • Memory: 64 GiB memory • Disk: Each disk of size 0.5 TiB or 2 TiB or 4 TiB storage • OSD: 3 OSDs in three different availability zones of Azure
Mon	10 GiB storage per Mon on each node
Platform	OpenShift Container Platform 4.5 and later
Default storage class	**managed-premium**

1.2. SIZING AND SCALING

The initial cluster of 3 nodes can later be expanded to a maximum of 9 nodes that can support up to 27 disks (3 disks on each node). In case of more than 3 worker nodes, the distribution of the disks depends on OpenShift scheduling and available resources.

Expand the cluster in sets of three nodes to ensure that your storage is replicated, and to ensure you can use at least three availability zones.

NOTE

You can expand the storage capacity only in the increment of the capacity selected at the time of installation.

The following tables shows the supported configurations for Red Hat OpenShift Container Storage.

Table 1.1. Initial configuration across 3 nodes

Disks	Disks per node	Total capacity	Usable storage capacity
0.5 TiB	1	1.5 TiB	0.5 TiB

Disks	Disks per node	Total capacity	Usable storage capacity
2 TiB	1	6 TiB	2 TiB
4 TiB	1	12 TiB	4 TiB

Table 1.2. Expanded configuration of up to 9 nodes

Disk size (N)	Maximum disks per node	Maximum total capacity (= 27 disks x N)	Maximum usable storage capacity
0.5 TiB	3	13.5 TiB	4.5 TiB
2 TiB	3	54 TiB	18 TiB
4 TiB	3	108 TiB	36 TiB

1.3. SUPPORTED WORKLOAD TYPES

Red Hat OpenShift Container Storage provides storage appropriate for a number of workload types.

Block storage is suitable for databases and other low-latency transactional workloads. Some examples of supported workloads are Red Hat OpenShift Container Platform logging and monitoring, and PostgreSQL.

Object storage is for video and audio files, compressed data archives, and the data used to train artificial intelligence or machine learning programs. In addition, object storage can be used for any application developed with a cloud-first approach.

File storage is for continuous integration and delivery, web application file storage, and artificial intelligence or machine learning data aggregation. Supported workloads include Red Hat OpenShift Container Platform registry and messaging using JBoss AMQ.

CHAPTER 2. DEPLOYING OPENSHIFT CONTAINER STORAGE ON MICROSOFT AZURE

You can deploy OpenShift Container Storage on Microsoft Azure installer-provisioned infrastructure (IPI). The deployment process consists of the following main parts:

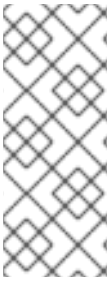
1. Install the OpenShift Container Storage Operator by following the instructions in Section 2.1, "Installing Red Hat OpenShift Container Storage Operator using the Operator Hub".
2. Create the OpenShift Container Storage service by following the instructions in Section 2.2, "Creating an OpenShift Container Storage service".
3. (Optional) Create a backing store over Azure Blob by following the instructions in Section 2.3, "Creating a new backing store".

2.1. INSTALLING RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR USING THE OPERATOR HUB

You can install Red Hat OpenShift Container Storage on Microsoft Azure platform using Red Hat OpenShift Container Platform Operator Hub. For information about hardware and software requirements, see Chapter 1, *Planning OpenShift Container Storage deployment on Microsoft Azure* .

Prerequisites

- Log in to OpenShift Container Platform cluster.
- You must have at least three worker nodes in the OpenShift Container Platform cluster.
- You must create a namespace called **openshift-storage** as follows:
    1. Click **Administration → Namespaces** in the left pane of the OpenShift Web Console.
    2. Click **Create Namespace**.
    3. In the Create Namespace dialog box, enter **openshift-storage** for Name and **openshift.io/cluster-monitoring=true** for Labels. This label is required to get the dashboards.
    4. Select **No restrictions** option for **Default Network Policy**.
    5. Click **Create**.



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=
```

Procedure

1. Click **Operators → OperatorHub** in the left pane of the OpenShift Web Console.

2. Click on **OpenShift Container Storage**.
   You can use the **Filter by keyword** text box or the filter list to search for OpenShift Container Storage from the list of operators.

3. On the **OpenShift Container Storage operator** page, click **Install**.

4. On the **Install Operator** page, ensure the following options are selected:

   a. Update Channel as **stable-4.4**

   b. Installation Mode as **A specific namespace on the cluster**

   c. Installed Namespace as **Operator recommended namespace PR openshift-storage**. If Namespace **openshift-storage** does not exist, it will be created during the operator installation.

   d. 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 **Installed Operators** page, click **ocs-operator**.

     iii. On the **Subscription Details** page, click the **Install Plan** link.

     iv. On the **InstallPlan Details** page, click **Preview Install Plan**

     v. Review the install plan and click **Approve**.

vi. Wait for the **Status** of the **Components** to change from **Unknown** to either **Created** or **Present**.

vii. Click **Operators → Installed Operators**

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

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

Verification steps

- Verify that **OpenShift Container Storage** Operator show the Status as **Succeeded**.

2.2. CREATING AN OPENSHIFT CONTAINER STORAGE SERVICE

You need to create a new OpenShift Container Storage service after you install OpenShift Container Storage operator.

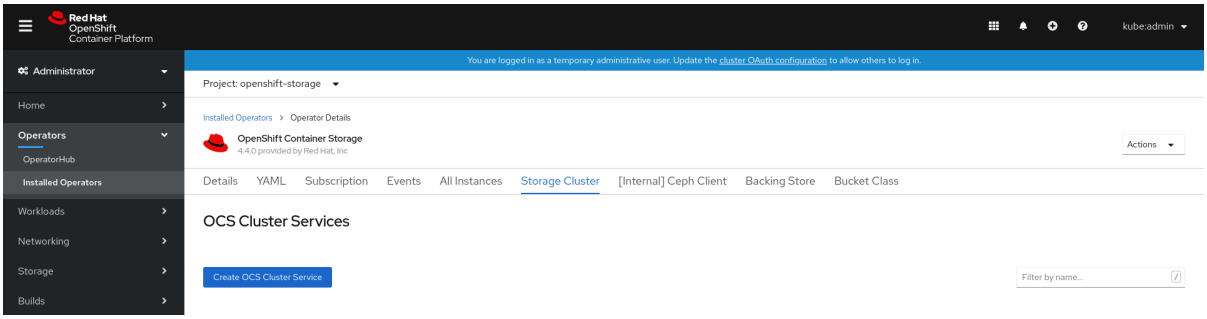
Prerequisites

- OpenShift Container Storage operator must be installed from the Operator Hub. For more information, see Installing OpenShift Container Storage Operator using the Operator Hub .

Procedure

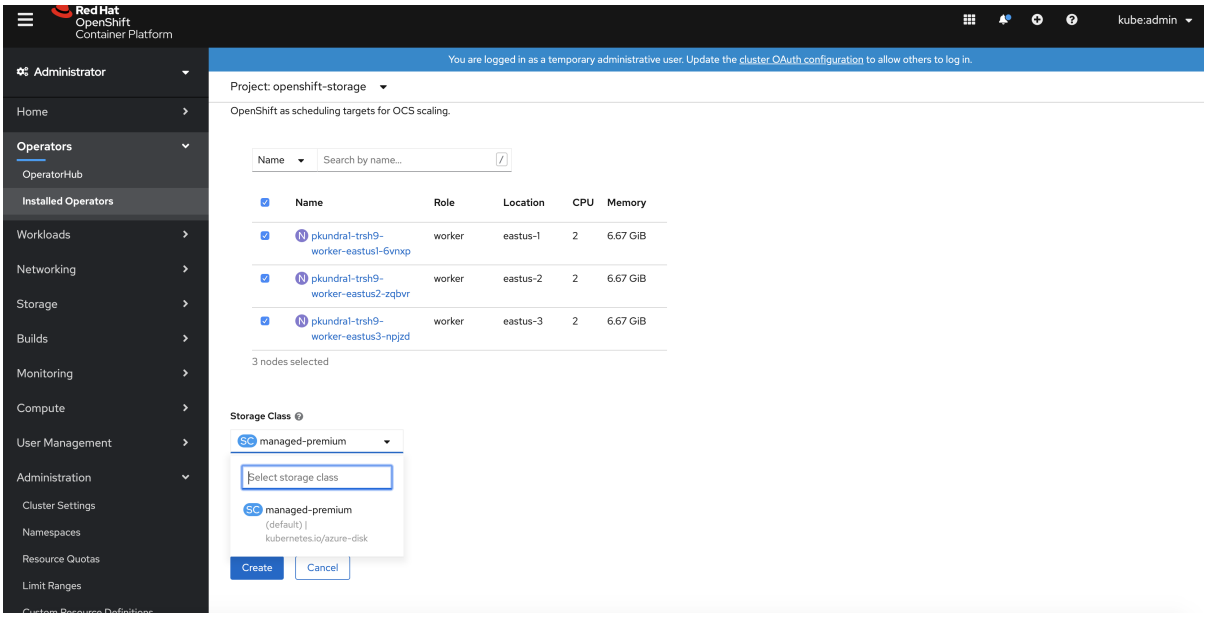
1. Click **Operators → Installed Operators** from the left pane of the OpenShift Web Console to view the installed operators.
2. On the Installed Operator page, select **openshift-storage** from the **Project** drop down list to switch to the **openshift-storage** project.
3. Click **OpenShift Container Storage** operator.
   OpenShift Container Storage operator creates a *OCSInitialization* resource automatically.
4. On the OpenShift Container Storage operator page, scroll right and click the **Storage Cluster** tab.

   Figure 2.1. OpenShift Container Storage Operator page



5. On the **OCS Cluster Services** page, click **Create OCS Cluster Service**.

Figure 2.2. Create New OCS Service page



6. On the **Create New OCS Service** page, perform the following:

   a. Select at least three worker nodes from the available list of nodes for the use of OpenShift Container Storage service. Ensure that the nodes are in different **Location**.

   b. **Storage Class** is set by default depending on the platform.
   managed-premium is the default storage for Azure.

   c. Select **OCS Service Capacity** from the drop down list.

NOTE

Once you select the initial storage capacity here, you can add more capacity only in this increment.

7. Click **Create**.
   The **Create** button is enabled only after you select three nodes. 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.

Verification steps

- To verify that OpenShift Container Storage is successfully installed, see Verifying OpenShift Container Storage deployment.

2.3. CREATING A NEW BACKING STORE

This procedure is not mandatory. However, it is recommended to perform this procedure.

When you install OpenShift Container Storage on Microsoft Azure platform, **noobaa-default-bucket-class** places data on **noobaa-default-backing-store** instead of Azure blob storage. Hence, to use OpenShift Container Storage Multicloud Object Gateway (MCG) managed object storage backed by Azure Blob storage, you need to perform the following procedure.

Before you begin

1. Log in to Azure web console.
2. Create Azure Blob storage account for MCG to store object data as described in Create a BlockBlobStorage account. Make sure to set Account kind as BlobStorage and connectivity method as public endpoint.
3. Locate access keys of the Blob storage account and note down the value for **key1** for later use.
4. Create a new Container within the new Blob storage account with public access level set as private.

Prerequisites

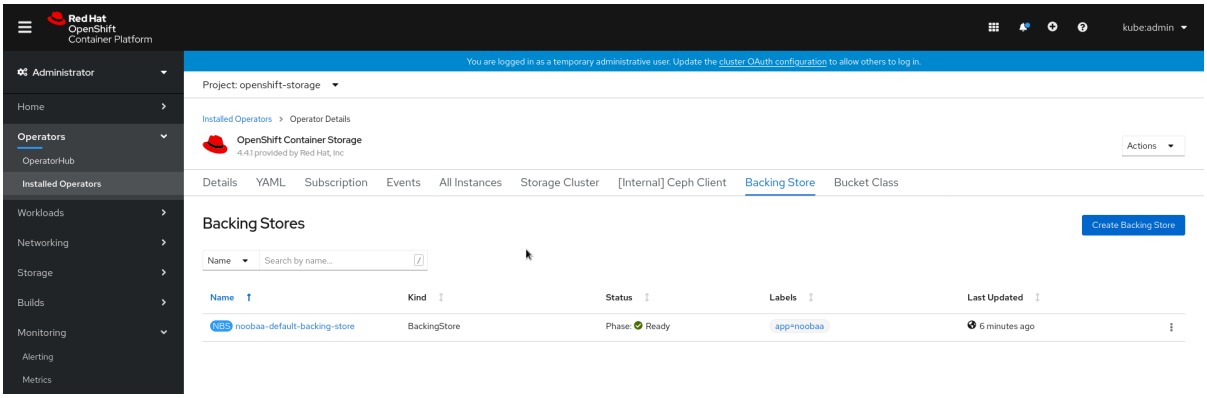
- Administrator access to OpenShift.

Procedure

To configure MCG to use Azure Blob storage account:

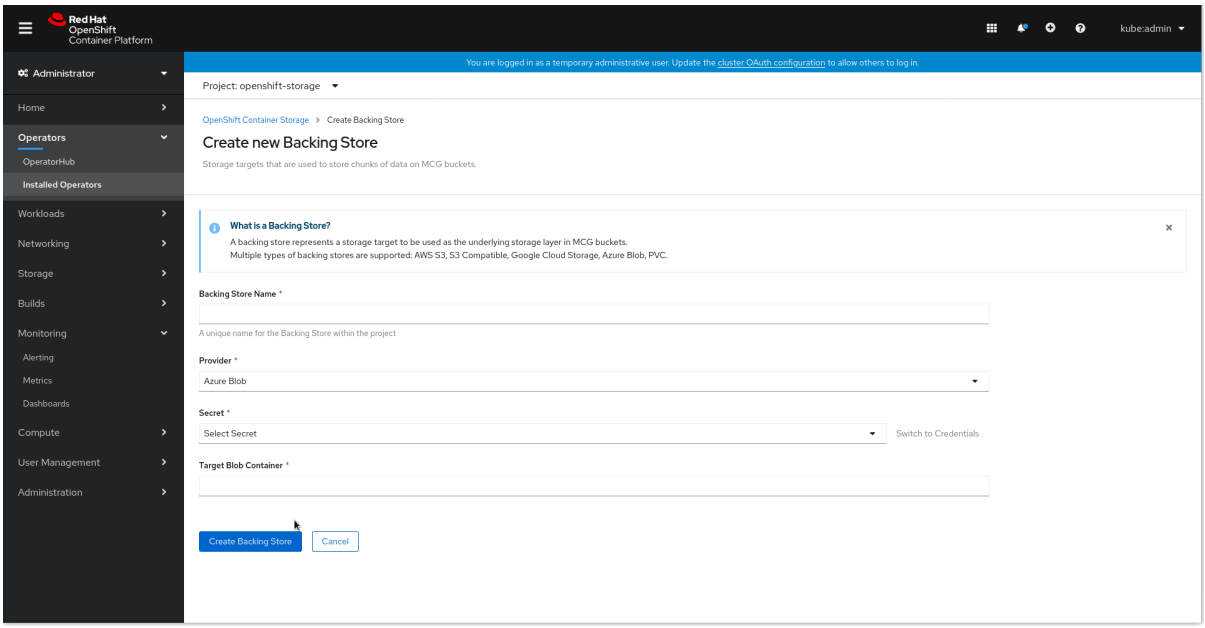
1. Log in to OpenShift Container Platform web console.
2. Click **Operators → Installed Operators** from the left pane of the OpenShift Web Console to view the installed operators.
3. Click **OpenShift Container Storage** Operator.
4. On the OpenShift Container Storage Operator page, scroll right and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

Figure 2.3. OpenShift Container Storage Operator page with backing store tab



5. Click **Create Backing Store**.

Figure 2.4. Create Backing Store page



6. On the Create New Backing Store page, perform the following:

   a. Enter a name for **Backing Store Name**.

   b. Select Azure Blob as the **Provider**.

   c. Click **Switch to Credentials**.

   d. Enter the **Account Name** of Azure Blob storage account you created earlier.

   e. Enter the value of **key1** of the Azure storage account you noted down earlier.

   f. Enter the name of the container that you created inside the Azure storage account for **Target Blob Container**. This allows you to create a connection that tells MCG that it can use this container for the system.

7. Click **Create Backing Store**.

8. In the OpenShift Container Platform web console, click **Installed Operators → OpenShift Container Storage → Bucket Class**.

9. Edit **noobaa-default-bucket-class** YAML specification field **spec: placementPolicy: tiers: - backingStores:** to use the newly created backing store instead of **noobaa-default-backing-store**.

Verification steps

1. Run the following command by using the MCG command line tool **noobaa** (from **mcg rpm** package) to verify that the Azure backing store that you created is in **Ready** state.

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

2. Verify that the output shows the default bucket class in **Ready** state and uses the expected backing store.

```
.
```

```
.
.
------------------
- Backing Stores -
------------------

NAME                          TYPE          TARGET-BUCKET
PHASE   AGE
noobaa-azure-backing-store            azure-blob      noobaabucketcontainer Ready   10m27s
noobaa-default-backing-store   s3-compatible
nb.1595507787728.apps.mbukatov20200723a.azure.qe.rh-ocs.com   Ready 1h58m20s

------------------
- Bucket Classes -
------------------

NAME                          PLACEMENT
PHASE   AGE
noobaa-default-bucket-class   {Tiers:[{Placement: BackingStores:[noobaa-azure-backing-
store]}]}   Ready   1h58m21s
```

2.4. VERIFYING OPENSHIFT CONTAINER STORAGE DEPLOYMENT

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

2.4.1. 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 amount of pods to expect for each component and how the amount of pods varies depending on the number of nodes and OSDs, see Table 2.1, "Pods corresponding to storage components for a three worker node 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 storage components for a three worker node cluster

Component	No. of pods	Name of the pod
Number of pods that you must see for the following components:		
OpenShift Container Storage Operator	1	**ocs-operator-***
Rook-ceph Operator	1	**rook-ceph-operator-***

Component	No. of pods	Name of the pod
Multicloud Object Gateway	4	• **noobaa-operator-*** • **noobaa-core-*** • **nooba-db-*** • **noobaa-endpoint-***
Mon	3	• **rook-ceph-mon-*** • **rook-ceph-mon-*** • **rook-ceph-mon-*** (on different nodes)
rook-ceph-mgr	1	**rook-ceph-mgr-*** (on storage node)
MDS	2	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** (2 pods on different storage nodes)
lib-bucket-provisioner	1	**lib-bucket-provisioner--*** (on any node)
Number of pods for CSI vary depending on the number of nodes selected as storage nodes (a minimum of 3 nodes)		

CSI	10	• **cephfs** (at least 5 pods)
		  ◦ **csi-cephfsplugin-*** (1 on each node where storage is consumed, that is, 3 pods on different nodes)
		  ◦ **csi-cephfsplugin-provisioner-*** (2 pods on different storage nodes if available)
		• **rbd** (at least 5 pods in total)
		  ◦ **csi-rbdplugin-*** (one on each node where storage is consumed, that is, 3 pods on different nodes)
		  ◦ **csi-rbdplugin-provisioner-*** (2 pods on different storage nodes if available)

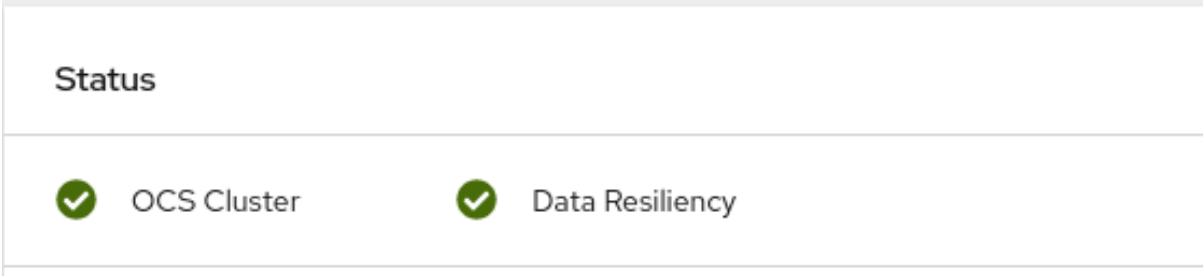
rook-ceph-drain-canary	3	**rook-ceph-drain-canary-*** (3 pods, that is, one on each storage node)
rook-ceph-crashcollector	3	**rook-ceph-crashcollector-*** (3 pods)

Component	No. of pods	Name of the pod
Number of OSDs vary depending on **Count** and **Replica** defined for each StorageDeviceSet in StorageCluster.		
OSD	6	• **rook-ceph-osd-*** (3 pods across different nodes) • **rook-ceph-osd-prepare-ocs-deviceset-*** (3 pods across different nodes)

2.4.2. Verifying the OpenShift Container Storage cluster is healthy

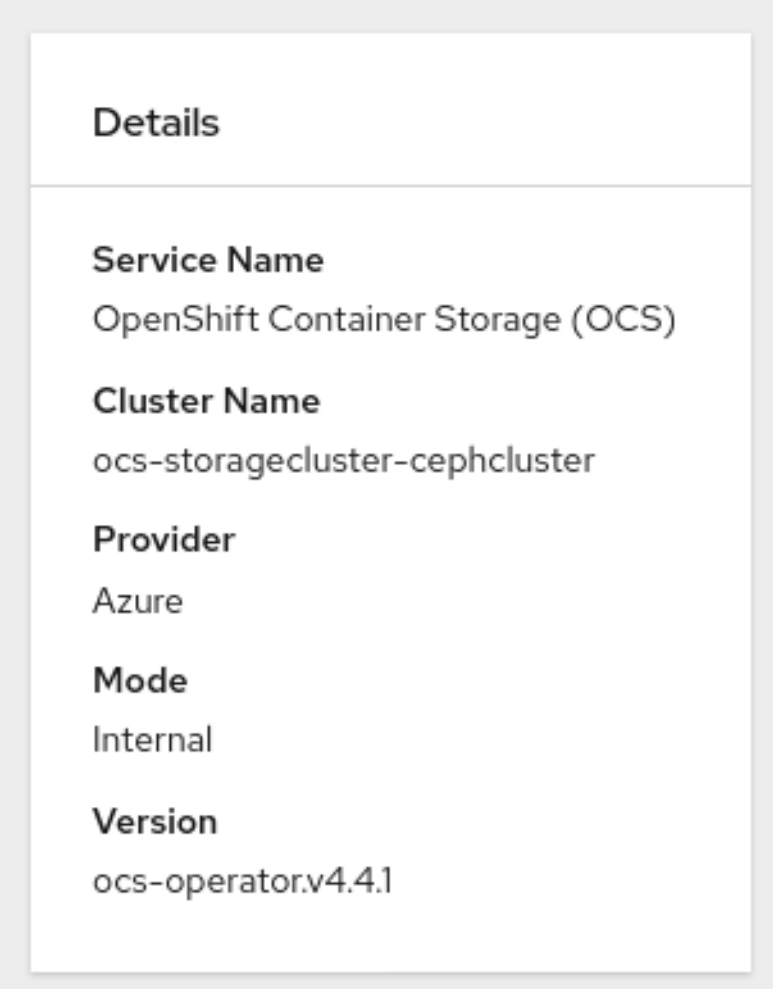
- Click **Home → Overview** from the left pane of the OpenShift Web Console and click **Persistent Storage** tab.
  In the **Status card**, verify that *OCS Cluster* has a green tick mark as shown in the following image:

  Figure 2.5. Health status card in Persistent Storage Overview Dashboard

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

Figure 2.6. Details card in Persistent Storage Overview Dashboard

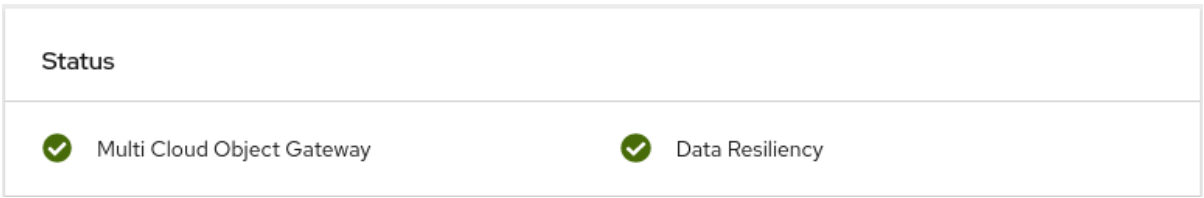


For more information on verifying the health of OpenShift Container Storage cluster using the persistent storage dashboard, see Monitoring OpenShift Container Storage .

2.4.3. Verifying the Multicloud Object Gateway is healthy

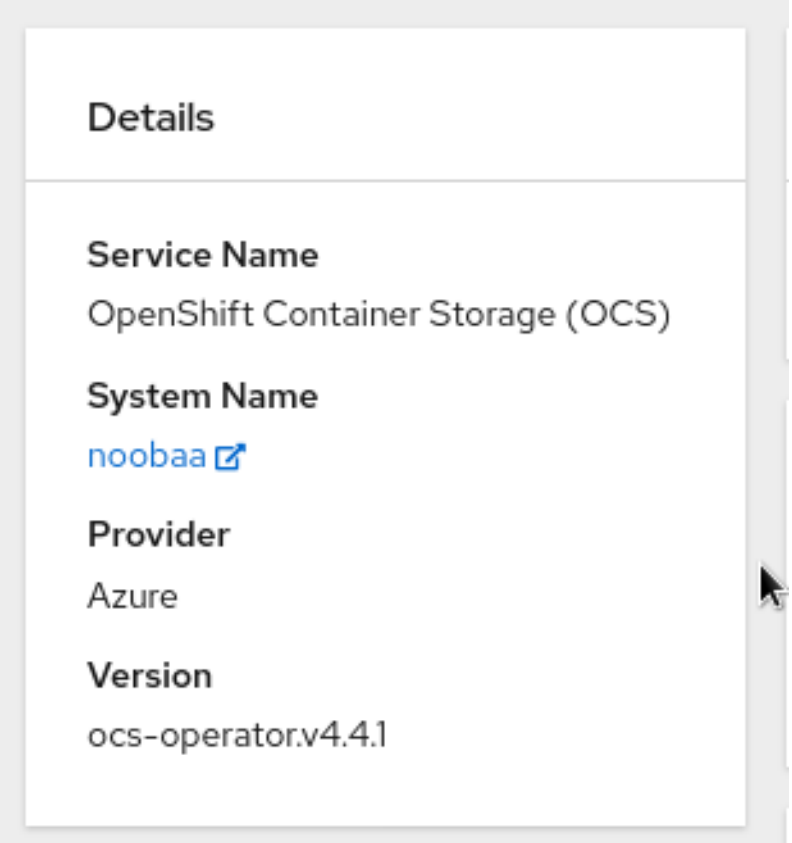
- Click **Home → Overview** from the left pane of the OpenShift Web Console and click the **Object Service** tab.
  In the **Status card**, verify that the Multicloud Object Gateway (MCG) storage displays a green tick icon as shown in following image:

  Figure 2.7. Health status card in Object Service Overview Dashboard

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

Figure 2.8. Details card in Object Service Overview Dashboard

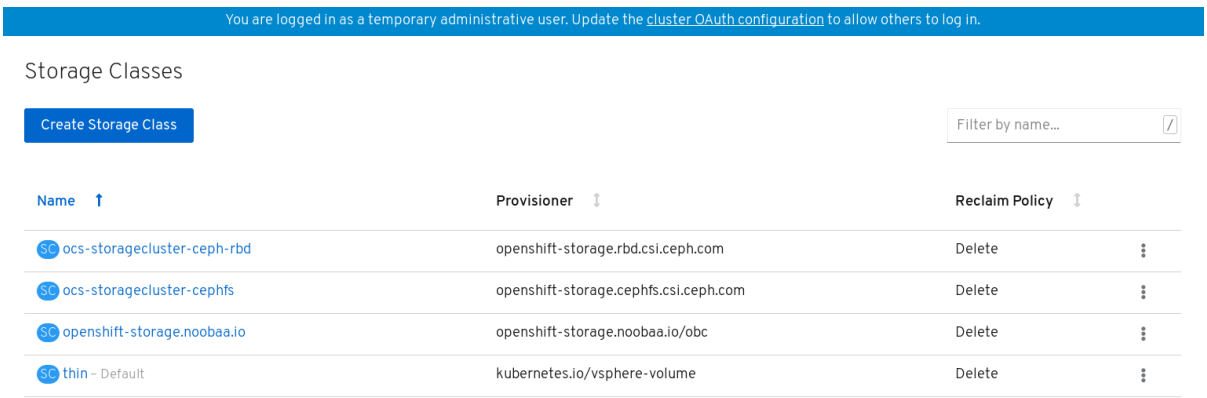


For more information on verifying the health of OpenShift Container Storage cluster using the object service dashboard, see Monitoring OpenShift Container Storage .

2.4.4. Verifying that the storage classes are created and listed

You can verify that the storage classes are created and listed as follows:

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



CHAPTER 3. UNINSTALLING OPENSHIFT CONTAINER STORAGE

Use the steps in this section to uninstall OpenShift Container Storage instead of the **Uninstall** option from the user interface.

Prerequisites

- Make sure that the OpenShift Container Storage cluster is in healthy state. The deletion might fail if some of the pods are not terminated successfully due to insufficient resources or nodes. In case the cluster is in unhealthy state, you should contact Red Hat Customer Support before uninstalling OpenShift Container Storage.
- Delete any applications that are consuming persistent volume claims (PVCs) or object bucket claims (OBCs) based on the OpenShift Container Storage storage classes and then delete PVCs and OBCs that are using OpenShift Container Storage storage classes.

Procedure

1. List the storage classes and take a note of the storage classes with the following storage class provisioners:
   - **openshift-storage.rbd.csi.ceph.com**
   - **openshift-storage.cephfs.csi.ceph.com**
   - **openshift-storage.noobaa.io/obc**

   For example:

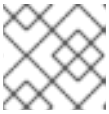
   ```
   $ oc get storageclasses
   NAME                          PROVISIONER                             RECLAIMPOLICY
   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
   managed-premium (default)     kubernetes.io/azure-disk                Delete
   WaitForFirstConsumer   true                   113m
   ocs-storagecluster-ceph-rbd   openshift-storage.rbd.csi.ceph.com      Delete
   Immediate              false                  95m
   ocs-storagecluster-cephfs     openshift-storage.cephfs.csi.ceph.com   Delete
   Immediate              false                  95m
   openshift-storage.noobaa.io   openshift-storage.noobaa.io/obc         Delete
   Immediate              false                  90m
   ```

2. Query for PVCs and OBCs that are using the storage class provisioners listed in the previous step.

   ```
   $ oc get pvc -o=jsonpath='{range .items[?(@.spec.storageClassName=="ocs-storagecluster-ceph-rbd")]}{"Name: "}{@.metadata.name}{" Namespace: "}{@.metadata.namespace}{" Labels: "}{@.metadata.labels}{"\n"}{end}' --all-namespaces|awk '! ( /Namespace: openshift-storage/ && /app:noobaa/ )'
   ```

   ```
   $ oc get pvc -o=jsonpath='{range .items[?(@.spec.storageClassName=="ocs-storagecluster-cephfs")]}{"Name: "}{@.metadata.name}{" Namespace: "}{@.metadata.namespace}{"\n"}{end}' --all-namespaces
   ```

```
$ oc get obc -o=jsonpath='{range .items[?(@.spec.storageClassName=="openshift-storage.noobaa.io")]}{"Name: "}{@.metadata.name}{" Namespace: "}{@.metadata.namespace}{"\n"}{end}' --all-namespaces
```



NOTE

Ignore any NooBaa PVCs in the **openshift-storage** namespace.

3. Follow these instructions to ensure that the PVCs listed in the previous step are deleted:

   a. Determine the pod that is consuming the PVC.

   b. Identify the controlling object such as a **Deployment**, **StatefulSet**, **DeamonSet**, **Job**, or a custom controller.
   Each object has a metadata field known as **OwnerReference**. This is a list of associated objects. The **OwnerReference** with the **controller** field set to **true** will point to controlling objects such as **ReplicaSet**, **StatefulSet**, **DaemonSet** and so on.

   c. Ensure that the object is safe to delete by asking the owner of the project and then delete it.

   d. Delete the PVCs and OBCs.

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

   If you have created any PVCs as a part of configuring the monitoring stack, cluster logging operator, or prometheus registry, then you must perform the clean up steps provided in the following sections as required:

   - Section 3.1, “Removing monitoring stack from OpenShift Container Storage”
   - Section 3.2, “Removing OpenShift Container Platform registry from OpenShift Container Storage”
   - Section 3.3, “Removing the cluster logging operator from OpenShift Container Storage”

4. List and note the backing local volume objects. If no results found, then skip step 8 & 9.

```
$ for sc in $(oc get storageclass|grep 'kubernetes.io/no-provisioner' |grep -E $(oc get storagecluster -n openshift-storage -o jsonpath='{
.items[*].spec.storageDeviceSets[*].dataPVCTemplate.spec.storageClassName}' | sed 's/
/|/g')| awk '{ print $1 }');
do
   echo -n "StorageClass: $sc ";
   oc get storageclass $sc -o jsonpath=" { 'LocalVolume: ' }{
.metadata.labels['local\.storage\.openshift\.io/owner-name'] } { '\n' }";
done
```

Example output

```
StorageClass: localblock  LocalVolume: local-block
```

5. Delete the **StorageCluster** object.

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

6. Delete the namespace and wait till the deletion is complete.

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

NOTE

You will need to switch to another project if openshift-storage was the active project.

For example

```
$ oc project default
```

7. Clean up the storage operator artifacts on each node.

```
$ 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 /var/lib/rook; done
```

Ensure you can see **removed directory*/var/lib/rook*** in the output.

Example output

```
Starting pod/ip-10-0-134-65us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
removed '/var/lib/rook/openshift-storage/log/ocs-deviceset-2-0-gk22s/ceph-volume.log'
removed directory '/var/lib/rook/openshift-storage/log/ocs-deviceset-2-0-gk22s'
removed '/var/lib/rook/openshift-storage/log/ceph-osd.2.log'
removed '/var/lib/rook/openshift-storage/log/ceph-volume.log'
removed directory '/var/lib/rook/openshift-storage/log'
removed directory '/var/lib/rook/openshift-storage/crash/posted'
removed directory '/var/lib/rook/openshift-storage/crash'
removed '/var/lib/rook/openshift-storage/client.admin.keyring'
removed '/var/lib/rook/openshift-storage/openshift-storage.config'
removed directory '/var/lib/rook/openshift-storage'
removed '/var/lib/rook/osd2/openshift-storage.config'
removed directory '/var/lib/rook/osd2'
removed directory '/var/lib/rook'

Removing debug pod ...
Starting pod/ip-10-0-155-149us-east-2computeinternal-debug ...
.
.
removed directory '/var/lib/rook'

Removing debug pod ...
Starting pod/ip-10-0-162-89us-east-2computeinternal-debug ...
.
.
```

```
removed directory '/var/lib/rook'

Removing debug pod ...
```

8. Delete the local volume created during the deployment and for each of the local volumes listed in step 4.
   For each of the local volumes, do the following:

   a. Set the variable **LV** to the name of the LocalVolume and variable **SC** to name of the StorageClass.

      For example

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

   b. List and note the devices to be cleaned up later.

      ```
      $ oc get localvolume -n local-storage $LV -o jsonpath='{
      .spec.storageClassDevices[*].devicePaths[*] }'
      ```

      Example output

      ```
      /dev/disk/by-id/nvme-Amazon_Elastic_Block_Store_vol078f5cdde09efc165 /dev/disk/by-id/nvme-Amazon_Elastic_Block_Store_vol0defc1d5e2dd07f9e /dev/disk/by-id/nvme-Amazon_Elastic_Block_Store_vol0c8e82a3beeb7b7e5
      ```

   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/ip-10-0-141-2us-east-2computeinternal-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/ip-10-0-144-55us-east-2computeinternal-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/ip-10-0-175-34us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
removed '/mnt/local-storage/localblock/nvme2n1'
removed directory '/mnt/local-storage/localblock'

Removing debug pod ...
```

9. Wipe the disks for each of the local volumes listed in step 4 so that they can be reused.

   a. List the storage nodes.

   ```
   $ oc get nodes -l cluster.ocs.openshift.io/openshift-storage=
   ```

   Example output

   ```
   NAME                                         STATUS   ROLES    AGE     VERSION
   ip-10-0-134-65.us-east-2.compute.internal    Ready    worker   4h45m   v1.17.1
   ip-10-0-155-149.us-east-2.compute.internal   Ready    worker   4h46m   v1.17.1
   ip-10-0-162-89.us-east-2.compute.internal    Ready    worker   4h45m   v1.17.1
   ```

   b. Obtain the node console and execute **chroot /host** command when the prompt appears.

   ```
   $ oc debug node/ip-10-0-134-65.us-east-2.compute.internal
   Starting pod/ip-10-0-134-65us-east-2computeinternal-debug ...
   To use host binaries, run `chroot /host`
   Pod IP: 10.0.134.65
   If you don't see a command prompt, try pressing enter.
   sh-4.2# chroot /host
   ```

   c. Store the disk paths gathered in step 8(ii) in the **DISKS** variable within quotes.

   ```
   sh-4.2# DISKS="/dev/disk/by-id/nvme-
   Amazon_Elastic_Block_Store_vol078f5cdde09efc165 /dev/disk/by-id/nvme-
   Amazon_Elasti_Block_Store_vol0defc1d5e2dd07f9e /dev/disk/by-id/nvme-
   Amazon_Elastic_Block_Store_vol0c8e82a3beeb7b7e5"
   ```

   d. Run **sgdisk --zap-all** on all the disks:

   ```
   sh-4.4# for disk in $DISKS; do sgdisk --zap-all $disk;done
   ```

   Example output

   ```
   Problem opening /dev/disk/by-id/nvme-
   Amazon_Elastic_Block_Store_vol078f5cdde09efc165 for reading! Error is 2.
   The specified file does not exist!
   Problem opening '' for writing! Program will now terminate.
   Warning! MBR not overwritten! Error is 2!
   Problem opening /dev/disk/by-id/nvme-
   Amazon_Elasti_Block_Store_vol0defc1d5e2dd07f9e for reading! Error is 2.
   ```

```
The specified file does not exist!
Problem opening '' for writing! Program will now terminate.
Warning! MBR not overwritten! Error is 2!
Creating new GPT entries.
GPT data structures destroyed! You may now partition the disk using fdisk or
other utilities.
```

NOTE

Ignore file-not-found warnings as they refer to disks that are on other machines.

e. Exit the shell and repeat for the other nodes.

```
sh-4.4# exit
exit
sh-4.2# exit
exit

Removing debug pod ...
```

10. Delete the storage classes with an **openshift-storage** provisioner listed in step 1.

```
$ oc delete storageclass <storageclass-name> --wait=true --timeout=5m
```

For example:

```
$ oc delete storageclass ocs-storagecluster-ceph-rbd ocs-storagecluster-cephfs openshift-
storage.noobaa.io --wait=true --timeout=5m
```

11. Unlabel the storage nodes.

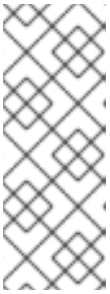
```
$ oc label nodes  --all cluster.ocs.openshift.io/openshift-storage-
$ oc label nodes  --all topology.rook.io/rack-
```

NOTE

You can ignore the warnings displayed for the unlabeled nodes such as **label <label> not found**.

12. Remove **CustomResourceDefinitions**.

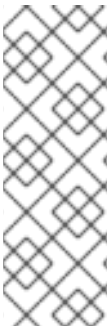
```
$ oc delete crd backingstores.noobaa.io bucketclasses.noobaa.io
cephblockpools.ceph.rook.io cephclusters.ceph.rook.io cephfilesystems.ceph.rook.io
cephnfses.ceph.rook.io cephobjectstores.ceph.rook.io cephobjectstoreusers.ceph.rook.io
noobaas.noobaa.io ocsinitializations.ocs.openshift.io
storageclusterinitializations.ocs.openshift.io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



NOTE

Uninstalling OpenShift Container Storage clusters on AWS deletes all the OpenShift Container Storage data stored on the target buckets, however, neither the target buckets created by the user nor the ones that were automatically created during the OpenShift Container Storage installation get deleted and the data that does not belong to OpenShift Container Storage remains on these target buckets.

13. To make sure that OpenShift Container Storage is uninstalled, verify that the openshift-storage namespace no longer exists and the storage dashboard no longer appears in the UI.



NOTE

While uninstalling OpenShift Container Storage, if namespace is not deleted completely and remains in **Terminating** state, perform the steps in the article https://access.redhat.com/solutions/3881901 to identify objects that are blocking the namespace from being terminated. OpenShift objects such as **Cephcluster**, **StorageCluster**, **NooBaa**, and **PVC** that have the finalizers might be the cause for the namespace to be in **Terminating** state. If PVC has a finalizer, force delete the associated pod to remove the finalizer.

3.1. REMOVING MONITORING STACK FROM OPENSHIFT CONTAINER STORAGE

Use this section to clean up 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. List the pods consuming the PVC.
In this example, the **alertmanagerMain** and **prometheusK8s** pods that were consuming the PVCs are in the **Terminating** state. You can delete the PVCs once these pods are no longer using OpenShift Container Storage PVC.

```
$ oc get pod,pvc -n openshift-monitoring
NAME                                          READY   STATUS      RESTARTS AGE
pod/alertmanager-main-0                       3/3     Terminating   0      10h
pod/alertmanager-main-1                       3/3     Terminating   0      10h
pod/alertmanager-main-2                       3/3     Terminating   0      10h
pod/cluster-monitoring-operator-84cd9df668-zhjfn   1/1     Running       0      18h
pod/grafana-5db6fd97f8-pmtbf                  2/2     Running       0      10h
pod/kube-state-metrics-895899678-z2r9q        3/3     Running       0      10h
pod/node-exporter-4njxv                       2/2     Running       0      18h
pod/node-exporter-b8ckz                       2/2     Running       0      11h
pod/node-exporter-c2vp5                       2/2     Running       0      18h
pod/node-exporter-cq65n                       2/2     Running       0      18h
pod/node-exporter-f5sm7                       2/2     Running       0      11h
pod/node-exporter-f852c                       2/2     Running       0      18h
pod/node-exporter-l9zn7                       2/2     Running       0      11h
pod/node-exporter-ngbs8                       2/2     Running       0      18h
pod/node-exporter-rv4v9                       2/2     Running       0      18h
pod/openshift-state-metrics-77d5f699d8-69q5x      3/3     Running       0      10h
pod/prometheus-adapter-765465b56-4tbxx        1/1     Running       0      10h
pod/prometheus-adapter-765465b56-s2qg2        1/1     Running       0      10h
pod/prometheus-k8s-0                          6/6     Terminating   1      9m47s
pod/prometheus-k8s-1                          6/6     Terminating   1      9m47s
pod/prometheus-operator-cbfd89f9-ldnwc        1/1     Running       0      43m
pod/telemeter-client-7b5ddb4489-2xfpz         3/3     Running       0      10h

NAME                                        STATUS  VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS               AGE
```

```
persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-0   Bound   pvc-
2eb79797-1fed-11ea-93e1-0a88476a6a64   40Gi       RWO           ocs-storagecluster-ceph-
rbd   19h
persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-1   Bound   pvc-
2ebeee54-1fed-11ea-93e1-0a88476a6a64   40Gi       RWO           ocs-storagecluster-ceph-
rbd   19h
persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-2   Bound   pvc-2ec6a9cf-
1fed-11ea-93e1-0a88476a6a64   40Gi       RWO           ocs-storagecluster-ceph-rbd   19h
persistentvolumeclaim/ocs-prometheus-claim-prometheus-k8s-0        Bound   pvc-3162a80c-
1fed-11ea-93e1-0a88476a6a64   40Gi       RWO           ocs-storagecluster-ceph-rbd   19h
persistentvolumeclaim/ocs-prometheus-claim-prometheus-k8s-1        Bound   pvc-
316e99e2-1fed-11ea-93e1-0a88476a6a64   40Gi       RWO           ocs-storagecluster-ceph-
rbd   19h
```

5. 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
```

3.2. 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: https://access.redhat.com/documentation/en-us/openshift_container_platform/4.4/html-single/registry/architecture-component-imageregistry

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
```

   - For Azure:

Before editing

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

After editing

```
.
.
storage:
.
.
```

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
```

3.3. 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 **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. 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 Curator and Modifying retention time for Prometheus metrics data in the OpenShift Container Platform documentation for details.

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

4.1. 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.

Follow the instructions in this section to configure OpenShift Container Storage as storage for the Container Image Registry. On Azure, it is not required to change the storage for the registry.

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.
- The **ocs-storagecluster-cephfs** storage class 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 OpenShift Web Console, click **Storage → Persistent Volume Claims**
   b. Set the **Project** to **openshift-image-registry**.
   c. Click **Create Persistent Volume Claim**
      i. Specify a **Storage Class** of **ocs-storagecluster-cephfs**.
      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**.

4.2. 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 intervals for this service. See the *Modifying retention time for Prometheus metrics data* sub section of Configuring persistent storage 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.
- The **ocs-storagecluster-ceph-rbd** storage class is available. In OpenShift Web Console, click **Storage → Storage Classes** to view available storage classes.

Procedure

1. In 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**.

   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

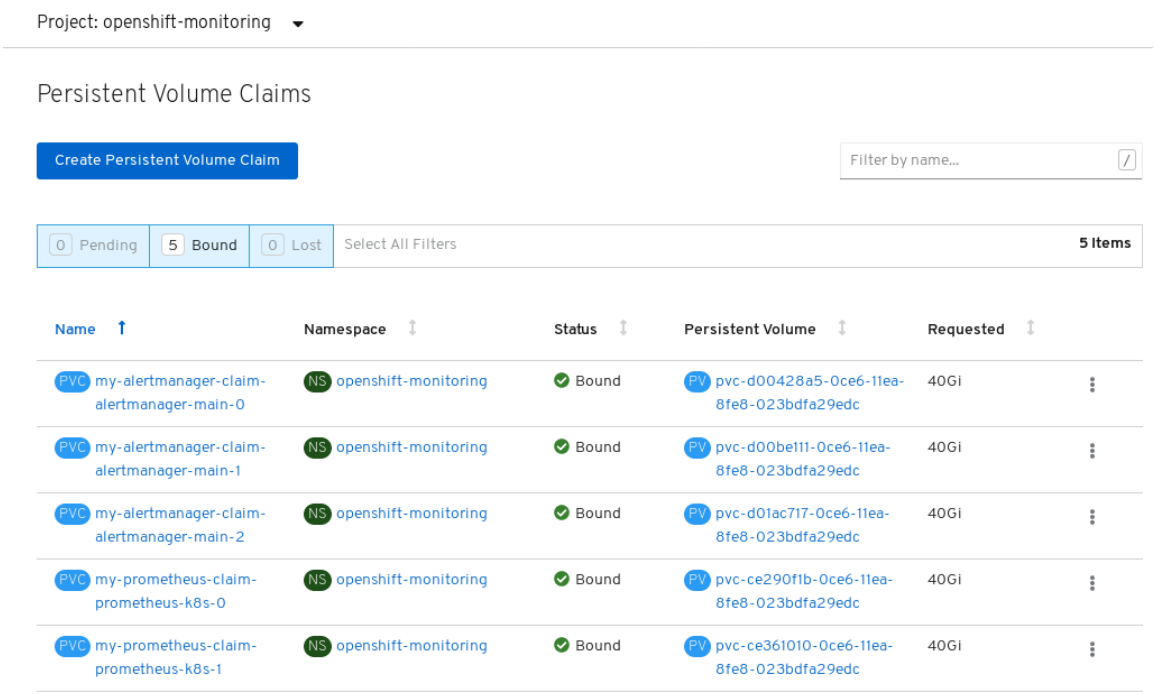
1. Verify that the Persistent Volume claims are bound to the pods.

   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

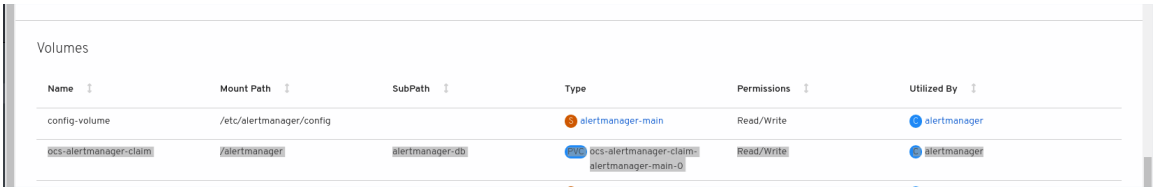


2. Verify that the new **alertmanager-main-*** pods appear with a state of **Running**.

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

   b. 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

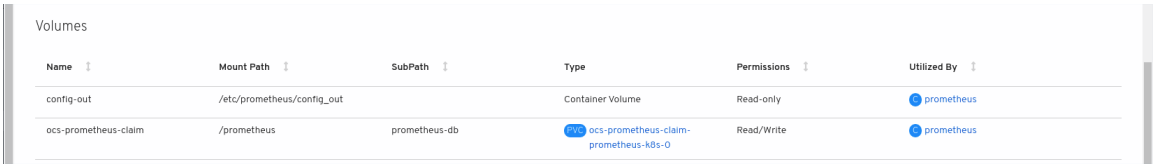


3. Verify that the new **prometheus-k8s-*** pods appear with a state of **Running**.

   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**.

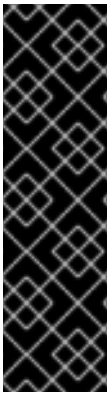
   Persistent Volume Claims attached to prometheus-k8s-* pod



4.3. 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 Configuring Curator in the OpenShift Container Platform documentation for details.

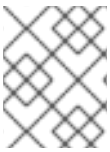
If you run out of storage space for these services, contact Red Hat Customer Support.

4.3.1. 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.

4.3.2. 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 code with the following:

```
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
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      replicas: 1
  curation:
    type: "curator"
```

```
    curator:
      schedule: "30 3 * * *"
  collection:
    logs:
      type: "fluentd"
      fluentd: {}
```

6. Click **Save**.

Verification steps

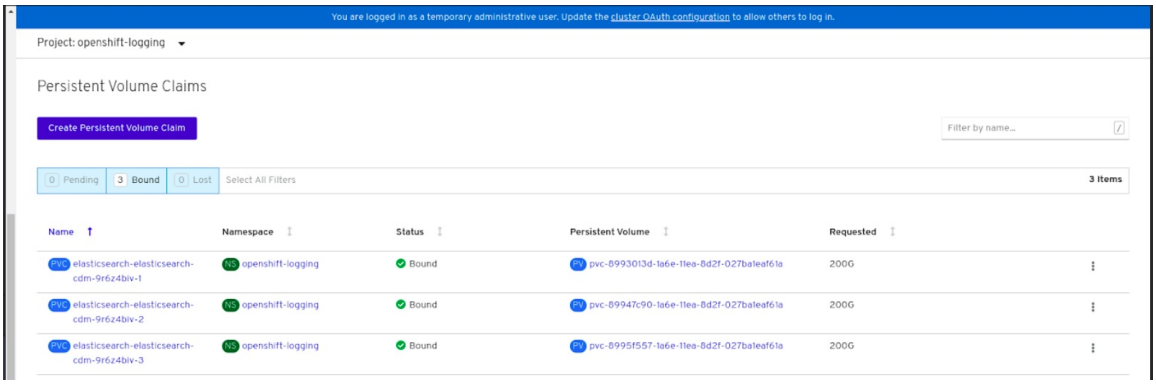
1. Verify that the Persistent Volume Claims are bound to the **elasticsearch** pods.

   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 4.1. Cluster logging created and bound



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

   a. Click **Workload → Pods**.

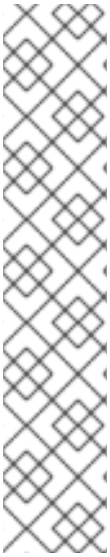
   b. Set the Project to **openshift-logging**.

   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.



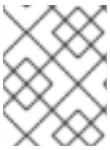
NOTE

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 cluster logging backed by Persistent Volume Claim, use the steps in Removing the cluster logging operator from OpenShift Container Storage .

CHAPTER 5. 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 **ocs-storagecluster-ceph-rbd** or **ocs-storagecluster-cephfs** 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 cannot resize 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 you 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 PersistenVolumeClaim Overview page.

CHAPTER 6. 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

Before you proceed to scale the storage nodes, refer to Section 1.1, "Requirements for installing OpenShift Container Storage on Microsoft Azure" to understand the node requirements for your specific OpenShift Container Storage instance.

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.

6.1. SCALING UP STORAGE BY ADDING CAPACITY TO YOUR OPENSHIFT CONTAINER STORAGE NODES ON AZURE INFRASTRUCTURE

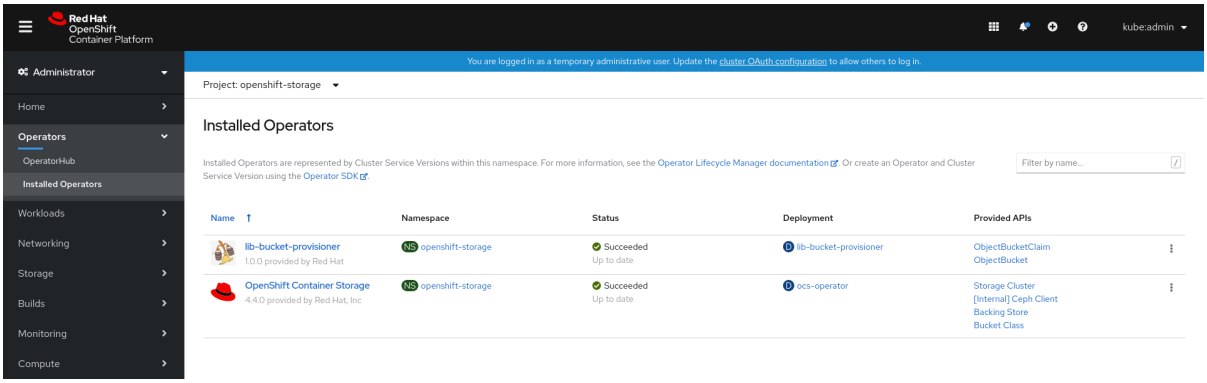
Use this procedure to add storage capacity and performance to your configured Red Hat OpenShift Container Storage worker nodes.

Prerequisites

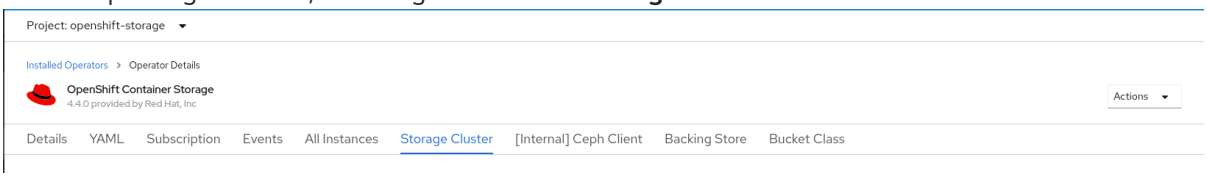
- A running OpenShift Container Storage Platform
- Administrative privileges on the OpenShift Web Console

Procedure

1. Navigate to the OpenShift Web Console.
2. Click on **Operators** on the left navigation bar.
3. Select **Installed Operators**.
4. In the window, click **OpenShift Container Storage** Operator:

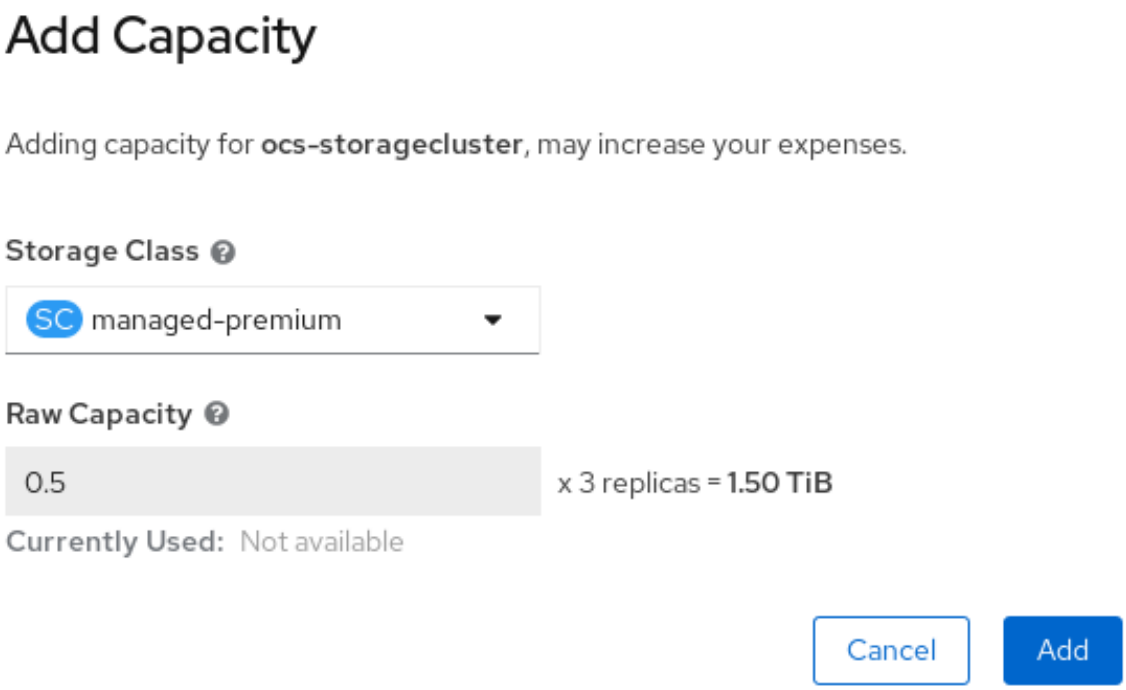


5. In the top navigation bar, scroll right and click **Storage Cluster** tab.



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

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



From this dialog box, you can set the requested additional capacity and the storage class. **Add capacity** shows the capacity selected at the time of installation and allows to add the capacity only in this increment. The storage class should be set to **managed-premium**.

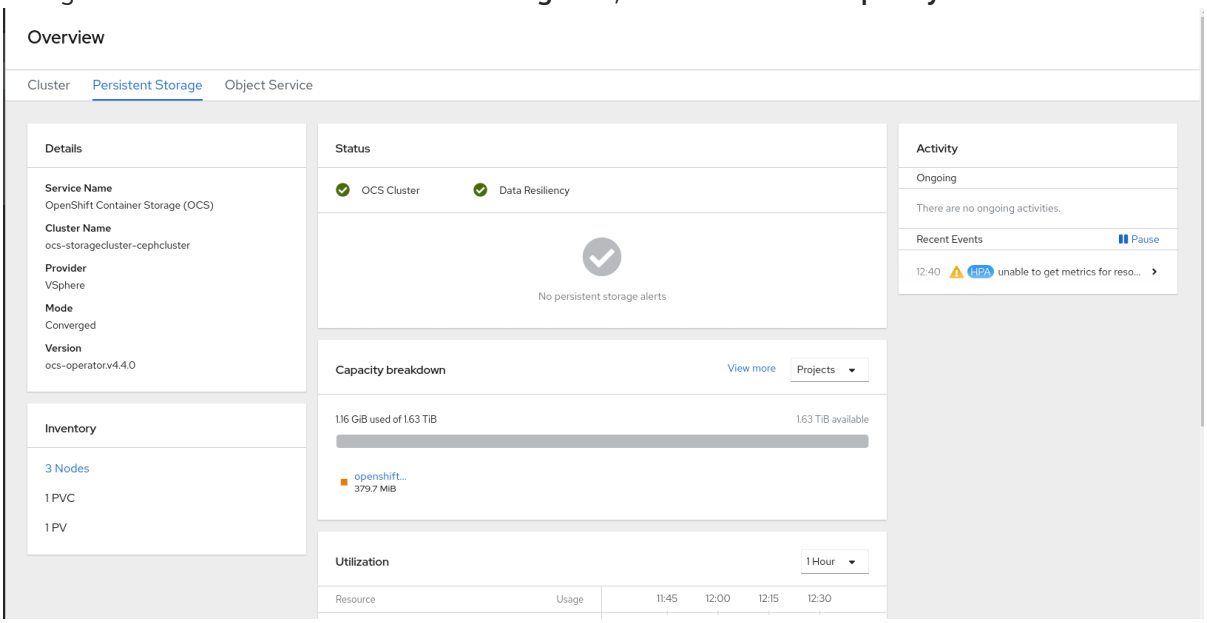
NOTE

The effectively provisioned capacity will be three times as much as what you see in the **Raw Capacity** field because OpenShift Container Storage uses a replica count of 3.

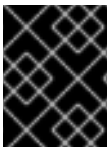
8. Click **Add**. You can see the status of the storage cluster after it reaches the **Ready** state. You might need to wait a couple of minutes after you see the **Ready** state.

Verification steps

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



2. Note that the capacity increases based on your selections.



IMPORTANT

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

6.2. 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

6.2.1. Adding a node on Azure installer-provisioned infrastructure

Prerequisites

- You must be logged into OpenShift Container Platform (OCP) cluster.

Procedure

1. Navigate to **Compute → Machine Sets**.
2. On the machine set where you want to add nodes, select **Edit Count**.
3. Add the amount of nodes, and click **Save**.
4. Click **Compute → Nodes** and confirm if the new node is in **Ready** state.

5. Apply the OpenShift Container Storage label to the new node.
    a. For the new node, **Action menu (⋮) → Edit Labels**.
    b. Add **cluster.ocs.openshift.io/openshift-storage** and 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 Section 6.2.2, “Verifying the addition of a new node” .

6.2.2. 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-***

6.2.3. 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 7. MULTICLOUD OBJECT GATEWAY

7.1. 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.

7.2. 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 here https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages

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

- Section 7.2.1, “Accessing the Multicloud Object Gateway from the terminal”
- Section 7.2.2, “Accessing the Multicloud Object Gateway from the MCG command-line interface”

7.2.1. 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.

7.2.2. 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 7.1. Example

If AWS S3 CLI is the application, the following command will list buckets in OCS:

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

7.3. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD

7.3.1. 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 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages

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**.

7.3.2. 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=http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-storage.svc.cluster.local:80
```

   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.
   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: http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-
storage.svc.cluster.local:80
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    signatureVersion: v4
    targetBucket: <RGW-bucket-name>
  type: s3-compatible
```

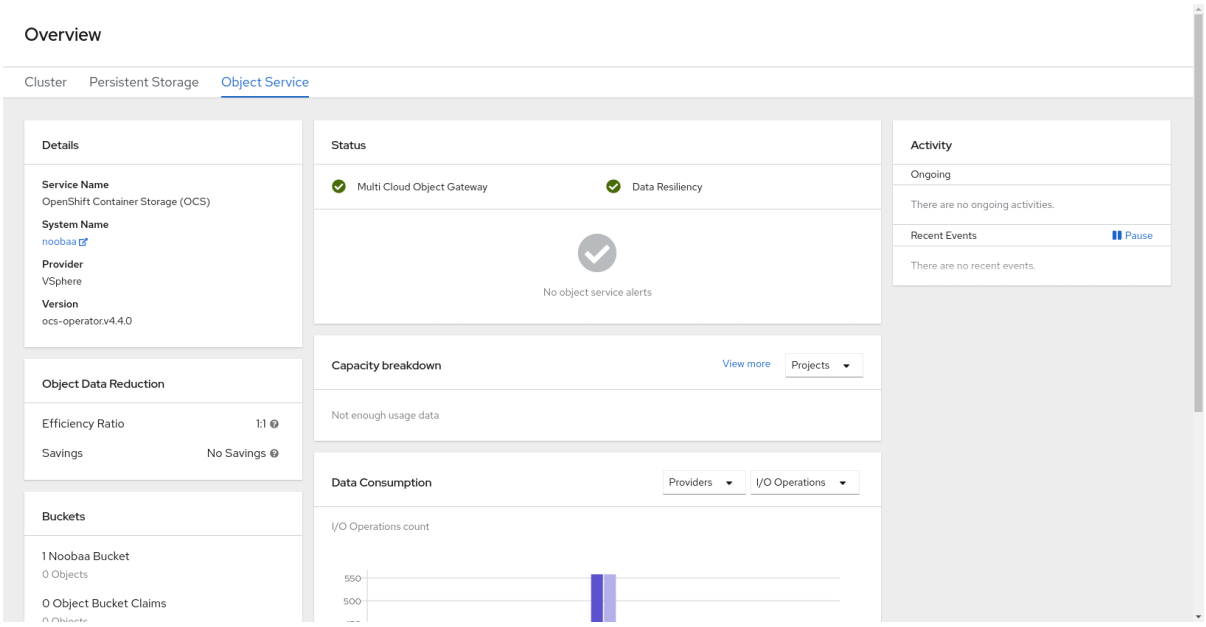
a. Replace **<backingstore-secret-name>** with the name of the secret that was created with **CephObjectStore** in the previous step.

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.

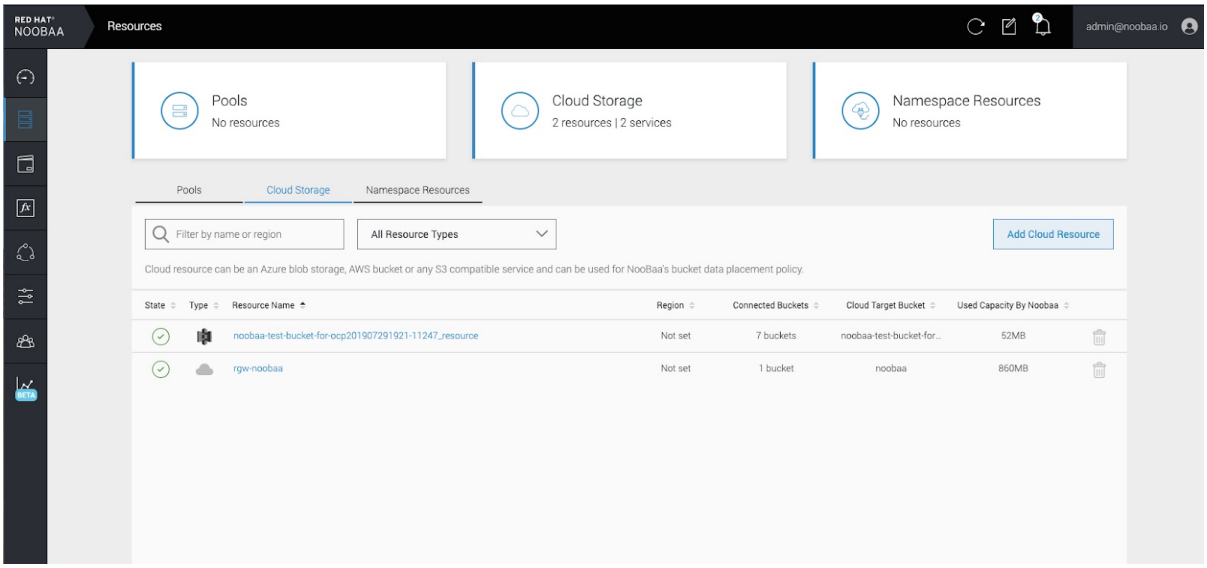
7.3.3. Adding storage resources for hybrid and Multicloud using the user interface

Procedure

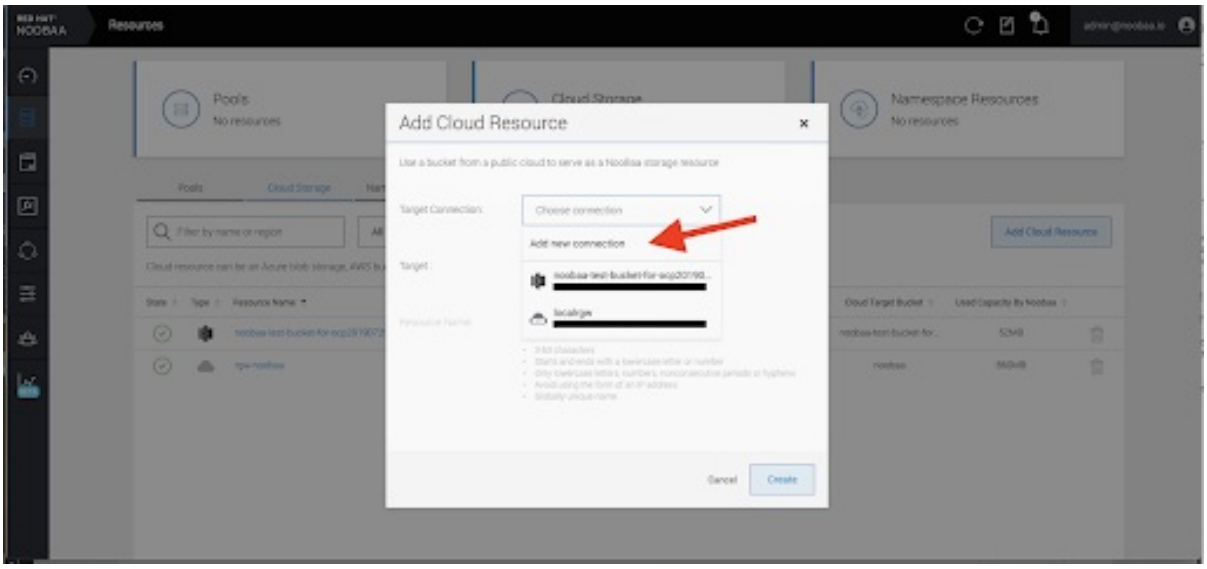
1. In your OpenShift Storage console, navigate to **Overview → Object Service →** select the **noobaa** link:



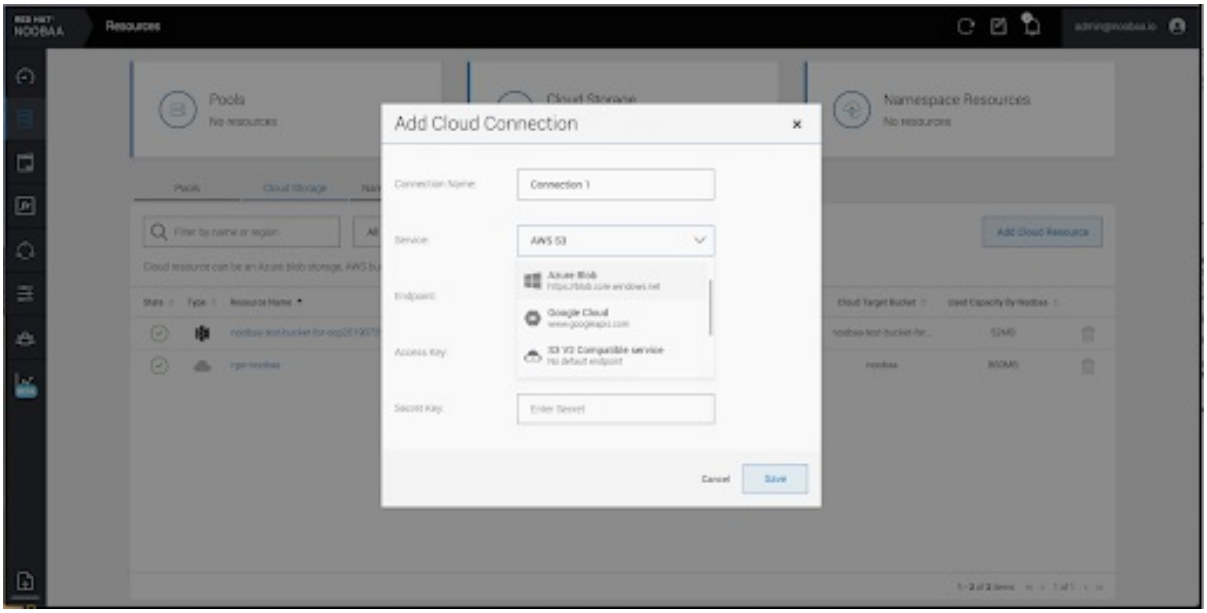
2. Select the **Resources** tab in the left, highlighted below. From the list that populates, select **Add Cloud Resource**:



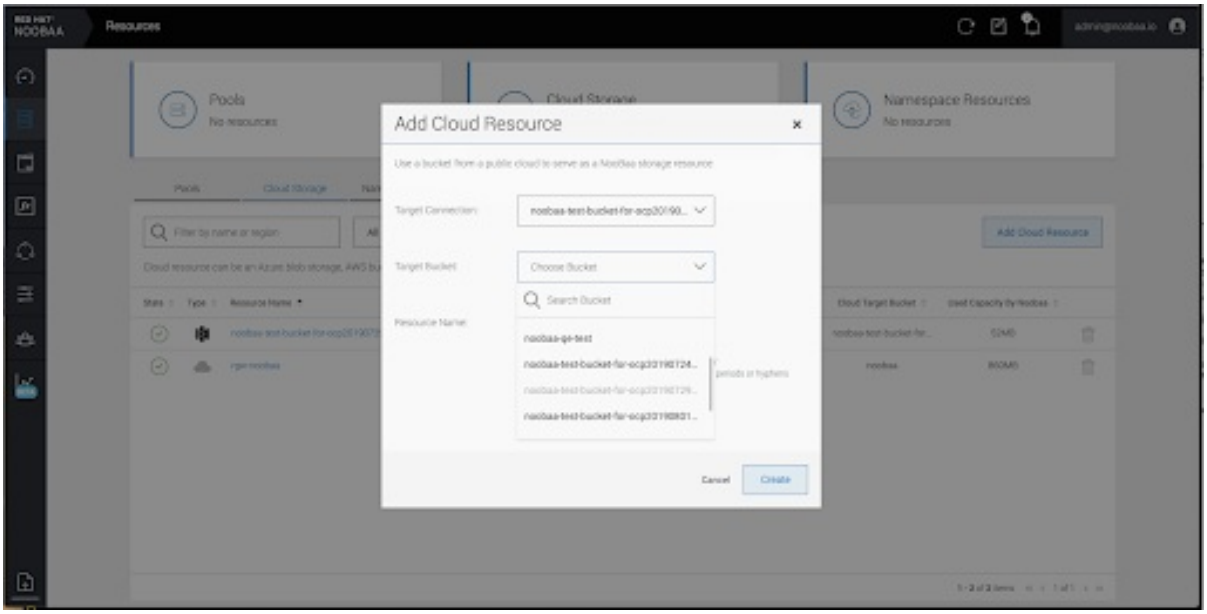
3. Select **Add new connection**:



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.

7.3.4. 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.

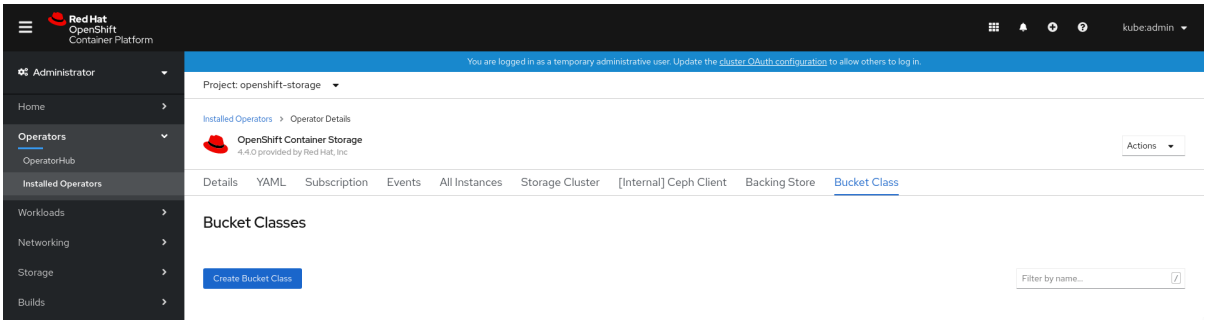
Procedure

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 7.1. OpenShift Container Storage Operator page with Bucket Class tab

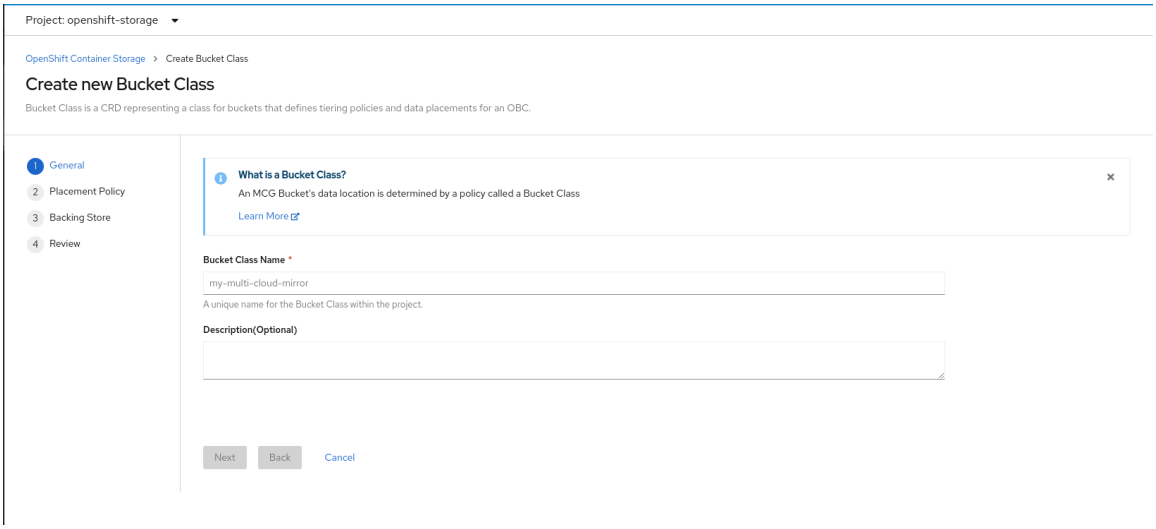


4. Click **Create Bucket Class**.

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

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

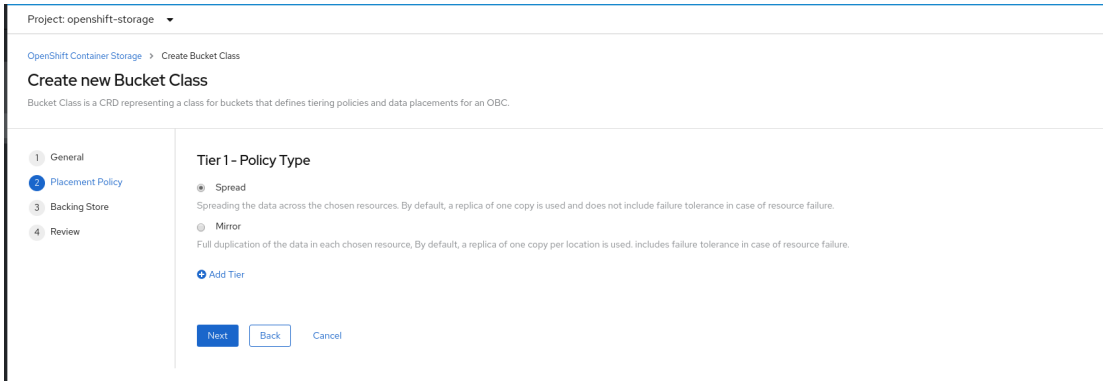
Figure 7.2. Create Bucket Class page



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

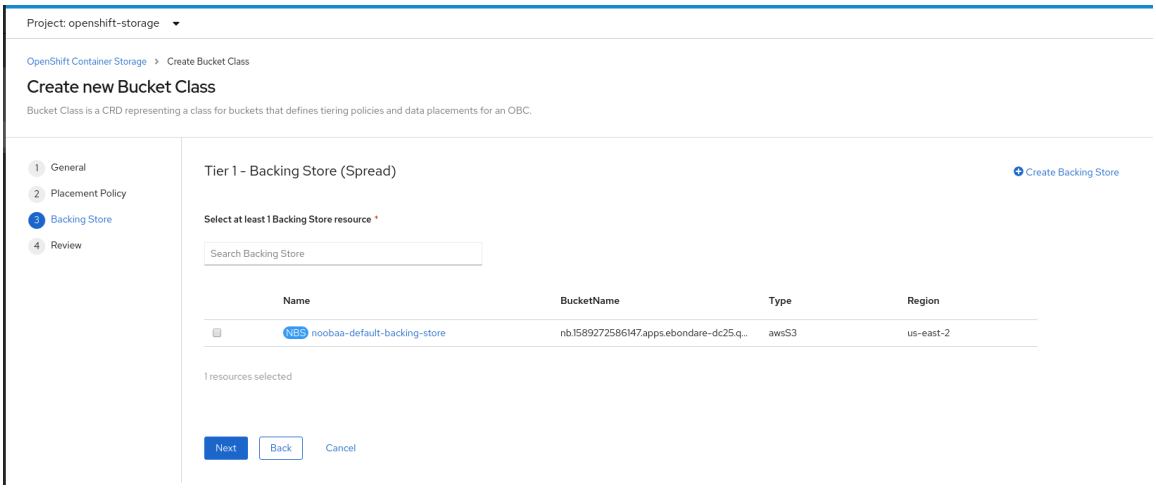
- **Spread** allows spreading of the data across the chosen resources.
- **Mirror** allows full duplication of the data across the chosen resources.
- Click **Add Tier** to add another policy tier.

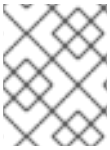
Figure 7.3. Tier 1 - Policy Type selection page



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 7.4. Tier 1 - Backing Store selection page



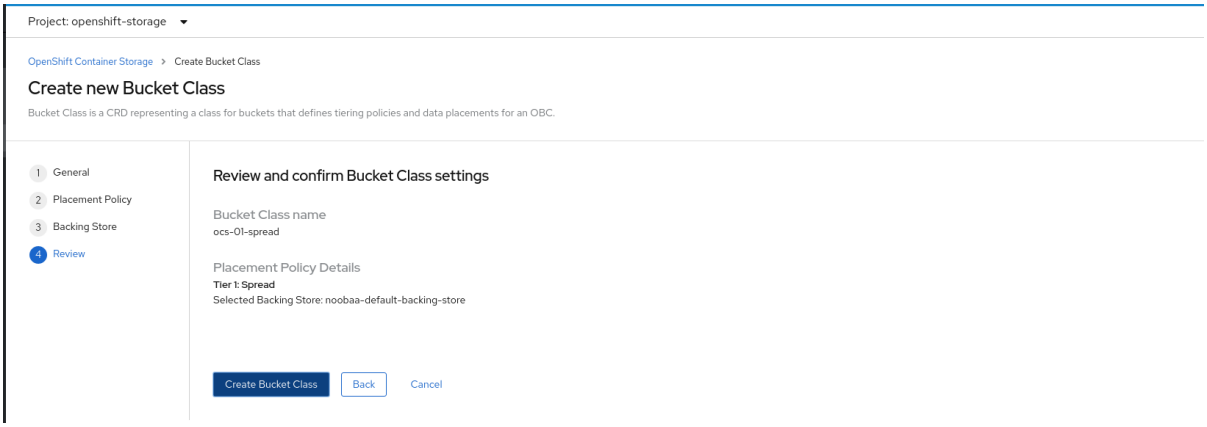


NOTE

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 7.5. 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.

7.3.5. Creating a new backing store

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

Prerequisites

- Administrator access to OpenShift.

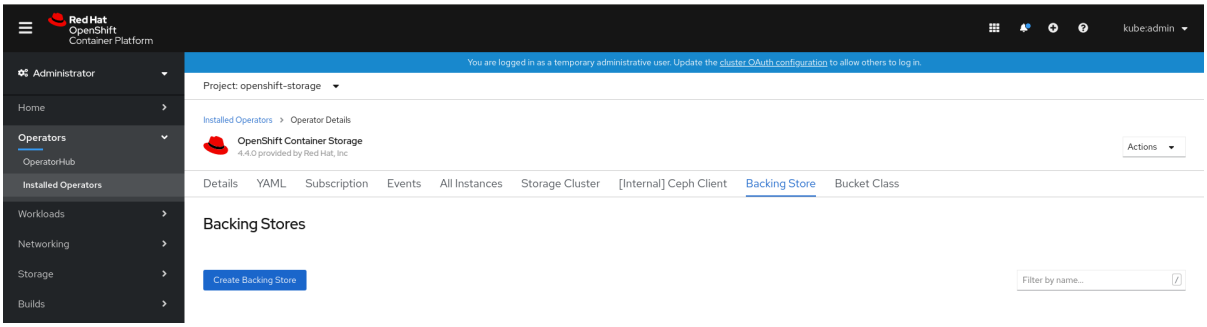
Procedure

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.

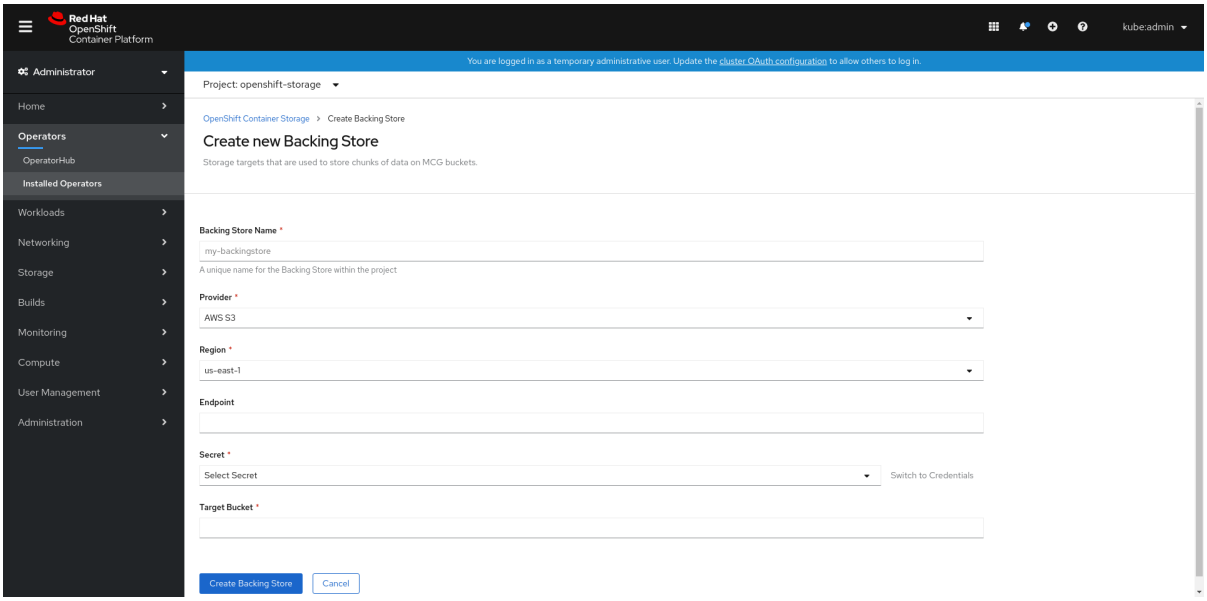
3. On the OpenShift Container Storage Operator page, scroll right and click the **Backing Store** tab.

Figure 7.6. OpenShift Container Storage Operator page with backing store tab



4. Click **Create Backing Store**.

Figure 7.7. 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 7.3.1, "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.

7.4. 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 7.3, "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 7.4.1, "Creating bucket classes to mirror data using the MCG command-line-interface"
- Section 7.4.2, "Creating bucket classes to mirror data using a YAML"
- Section 7.4.3, "Configuring buckets to mirror data using the user interface"

7.4.1. 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
```

7.4.2. 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
```

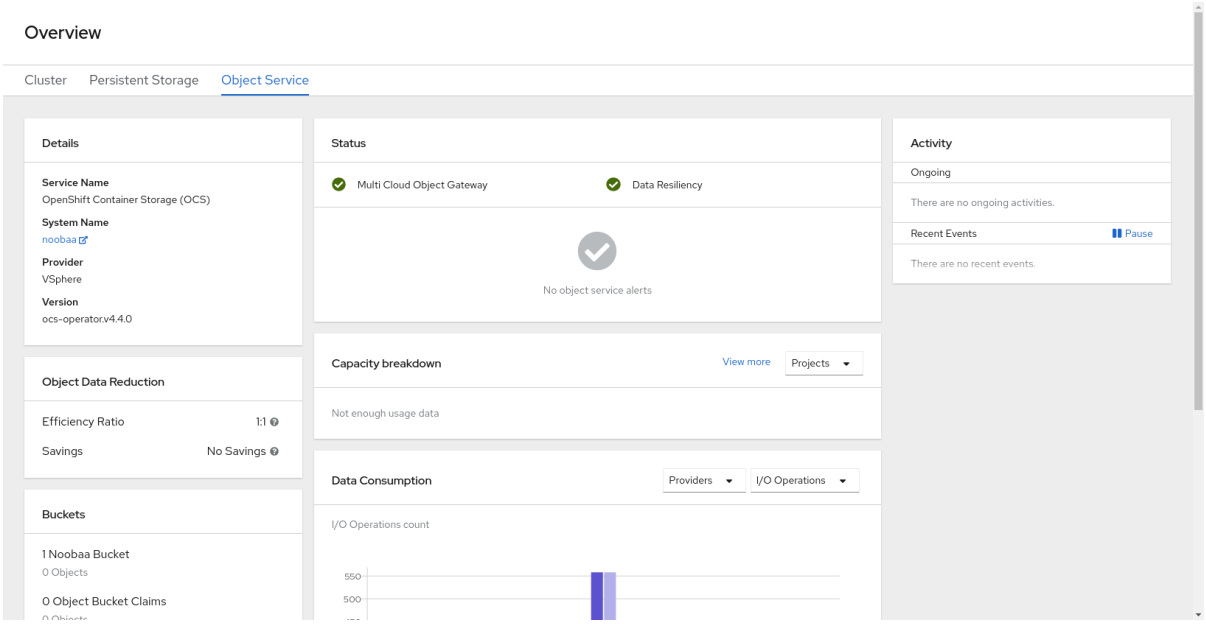
2. Add the following lines to your standard Object Bucket Claim (OBC):

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

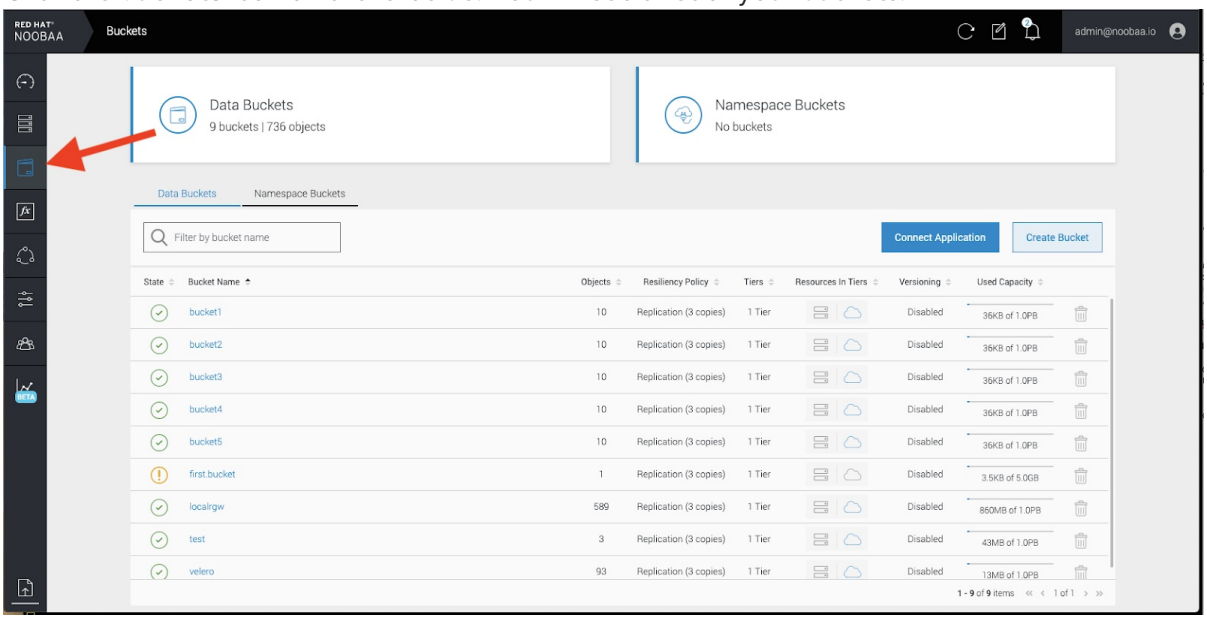
For more information about OBCs, see Section 7.6, “Object Bucket Claim”.

7.4.3. Configuring buckets to mirror data using the user interface

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

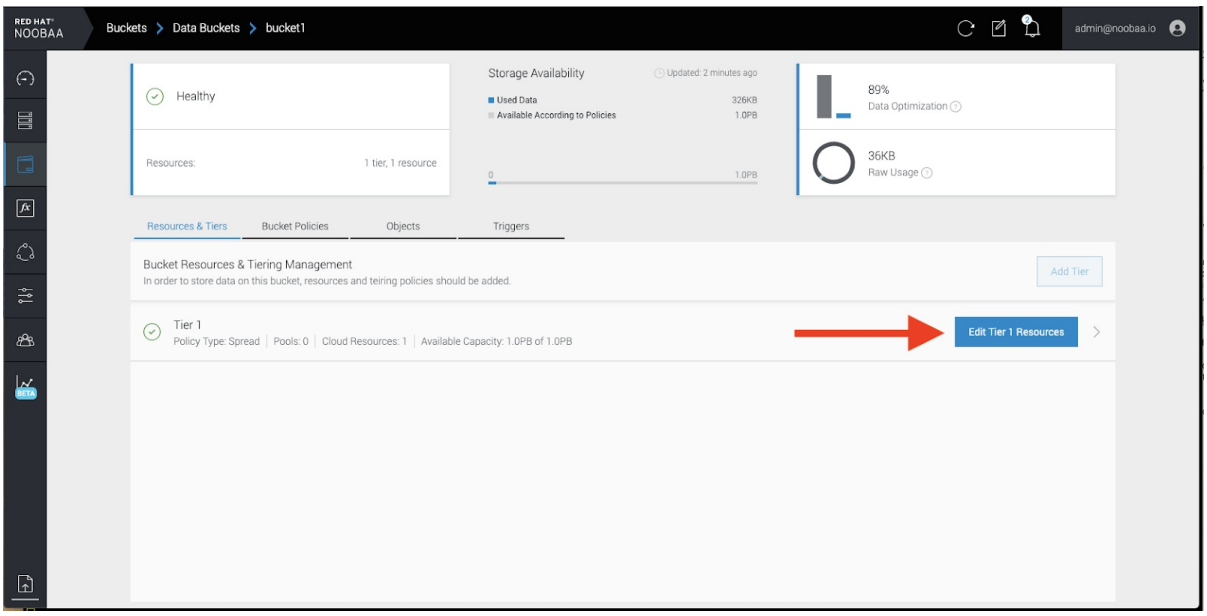


2. Click the **buckets** icon on the left side. You will see a list of your buckets:

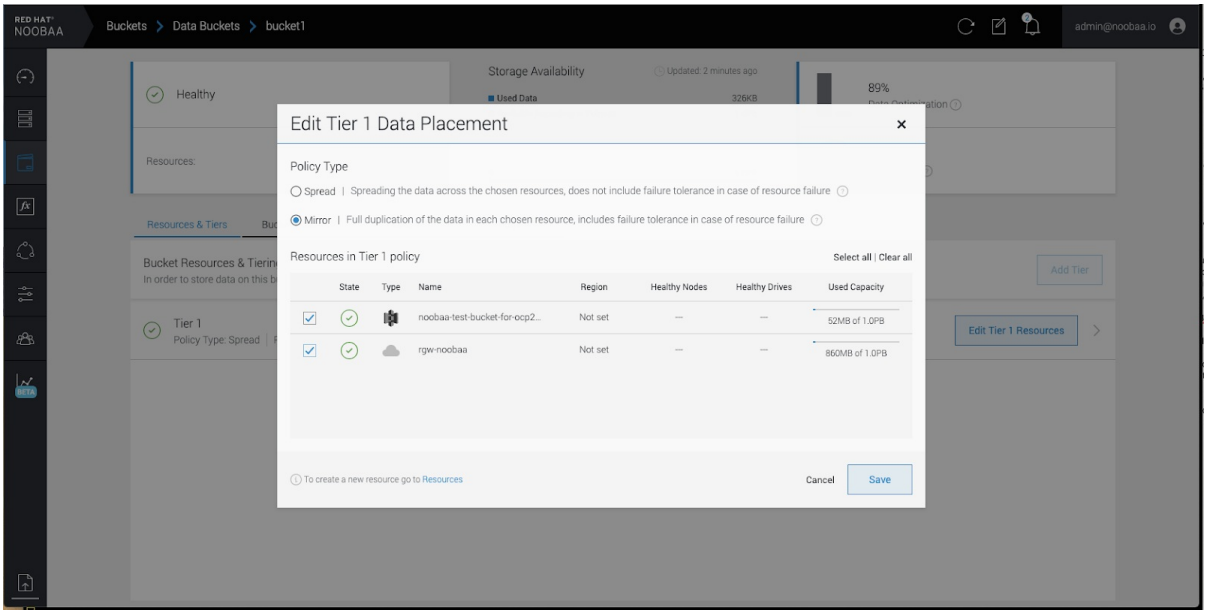


3. Click the bucket you want to update.

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.

7.5. 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.

7.5.1. 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 .

7.5.2. Using bucket policies

Prerequisites

- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see Section 7.2, “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 7.5.3, “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

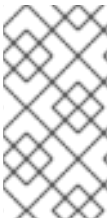
Replace ***BucketPolicy*** with the bucket policy JSON file

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.

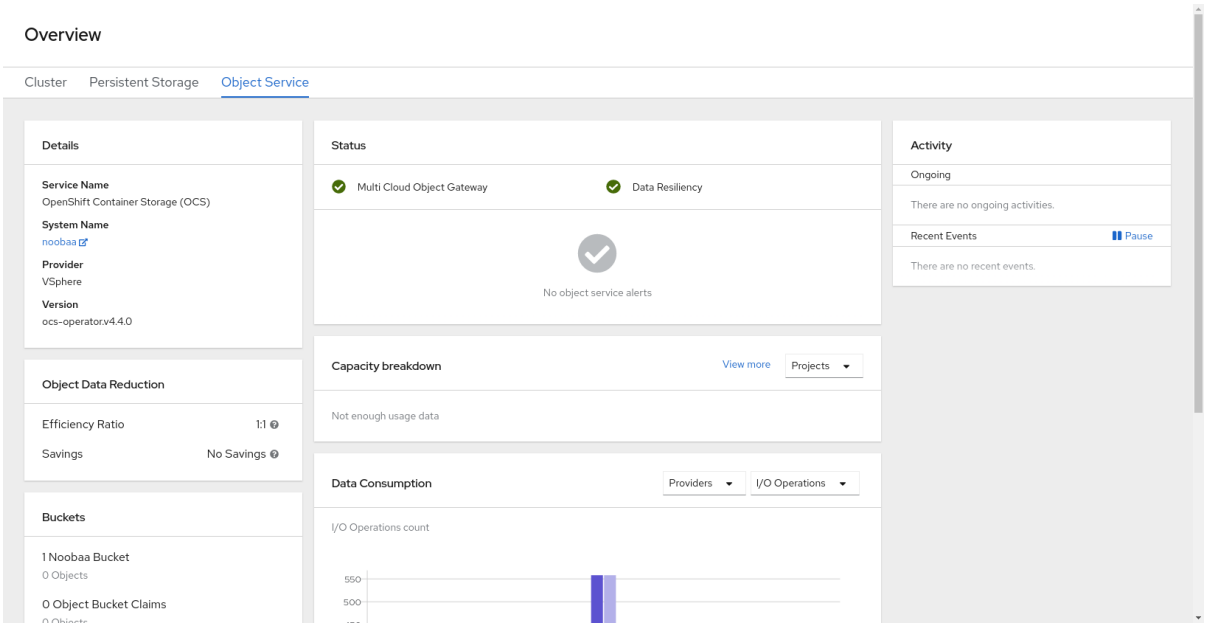
7.5.3. Creating an AWS S3 user in the Multicloud Object Gateway

Prerequisites

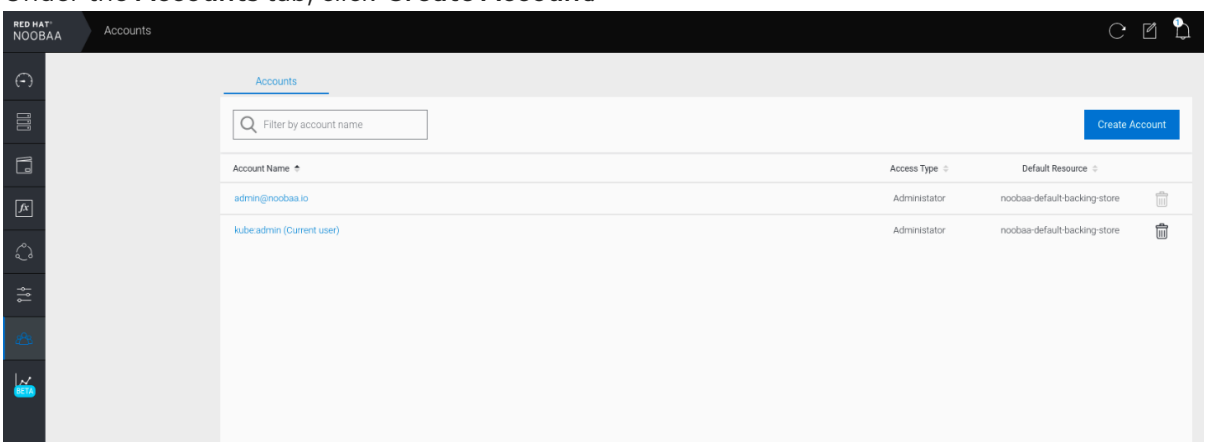
- A running OpenShift Container Storage Platform
- Access to the Multicloud Object Gateway, see Section 7.2, "Accessing the Multicloud Object Gateway with your applications"

Procedure

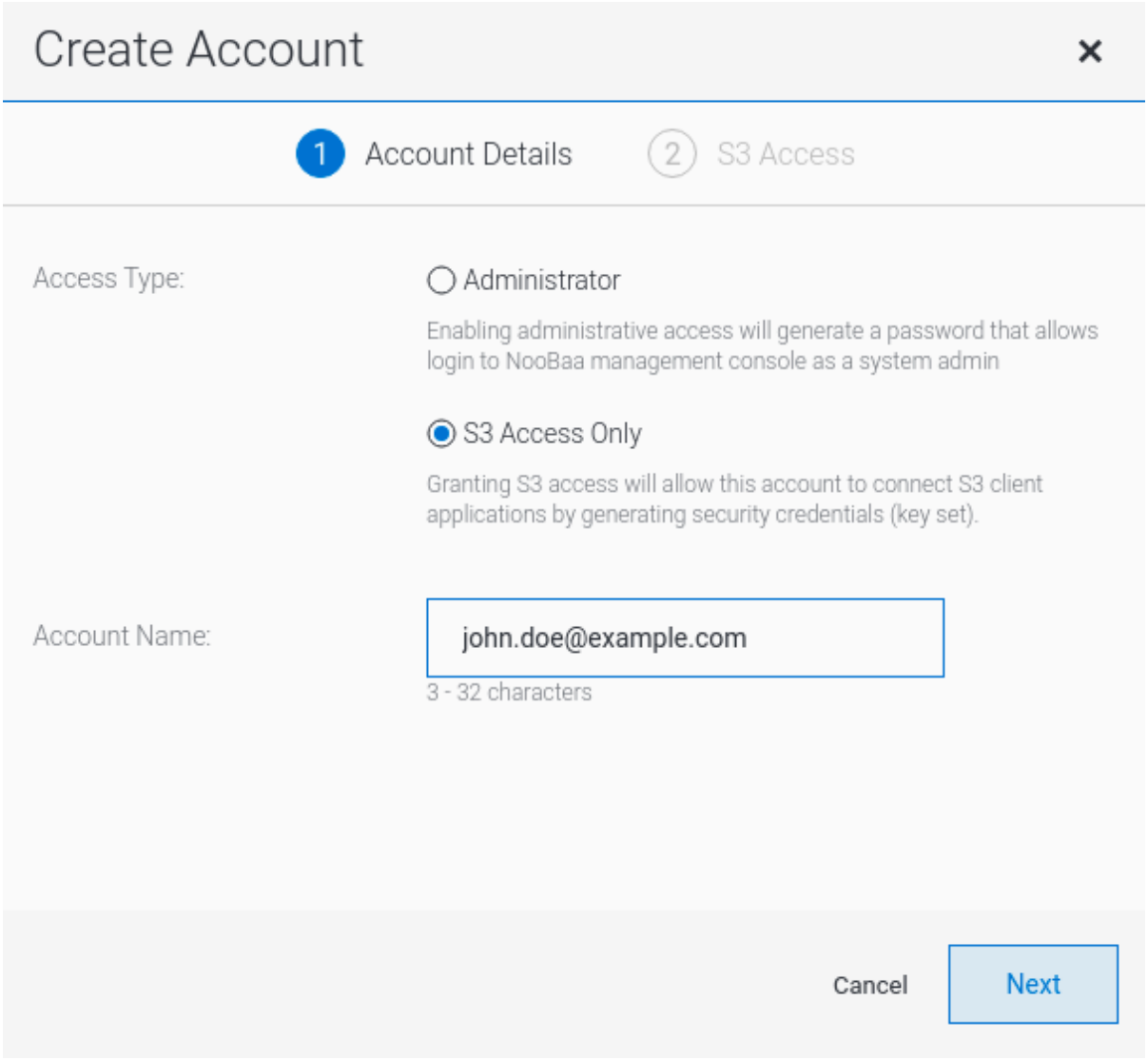
1. In your OpenShift Storage console, navigate to **Overview → Object Service →** select the **noobaa** link:



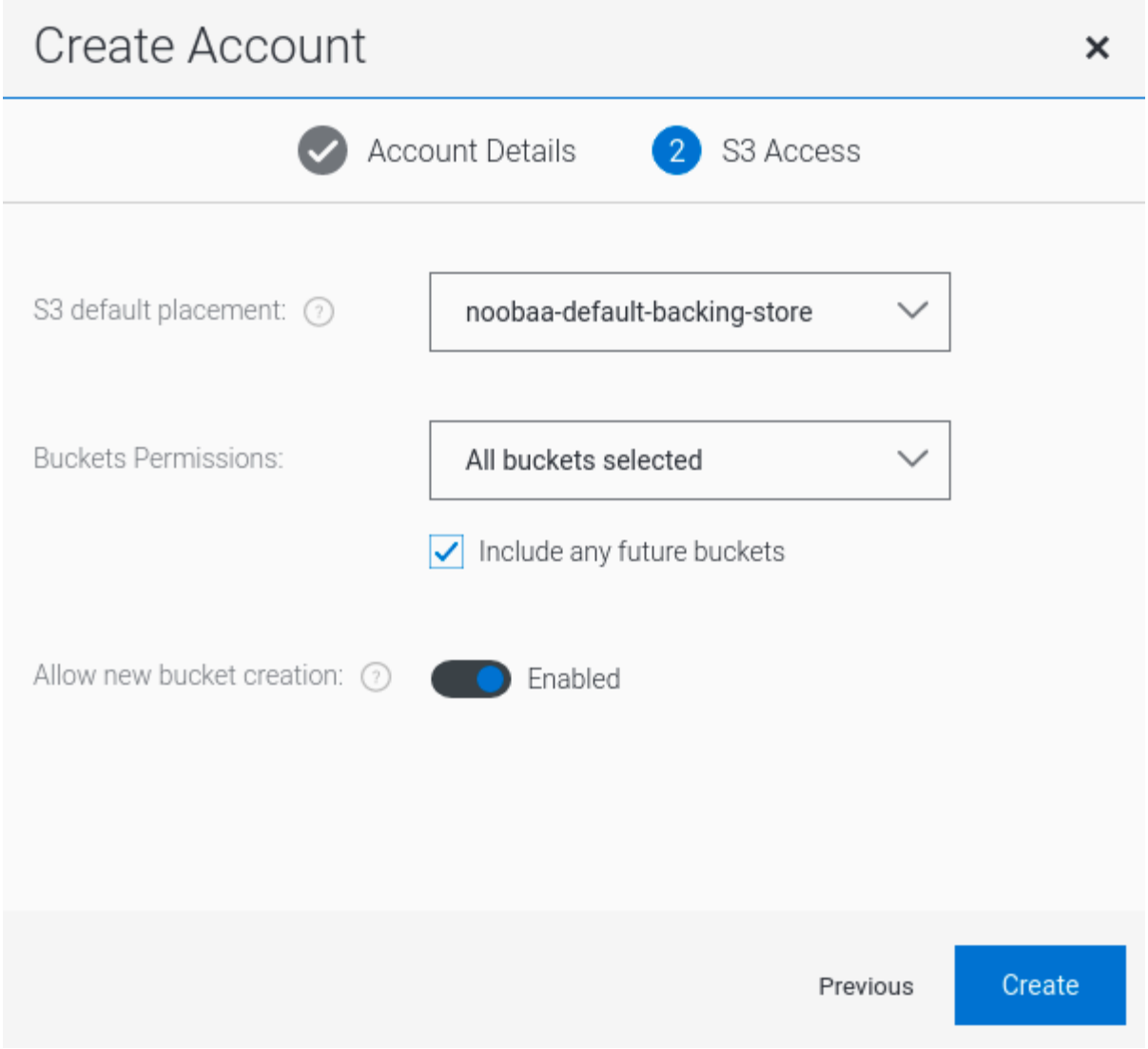
2. Under the **Accounts** tab, click **Create Account**:



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**:



7.6. 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 7.6.1, “Dynamic Object Bucket Claim”
- Section 7.6.2, “Creating an Object Bucket Claim using the command line interface”
- Section 7.6.3, “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.

7.6.1. 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: noobaa
```

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

7.6.2. 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.

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

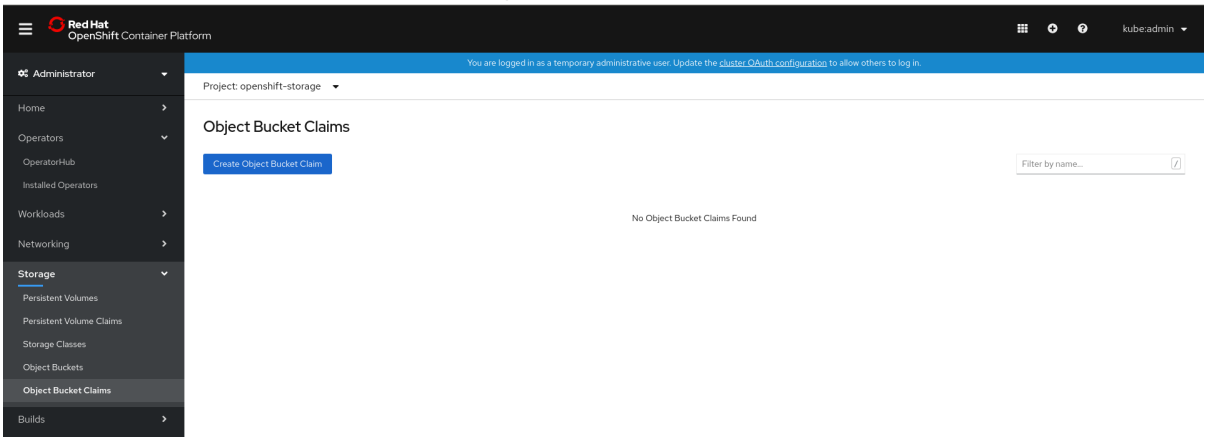
You can create an Object Bucket Claim (OBC) using the OpenShift Web Console.

Prerequisites

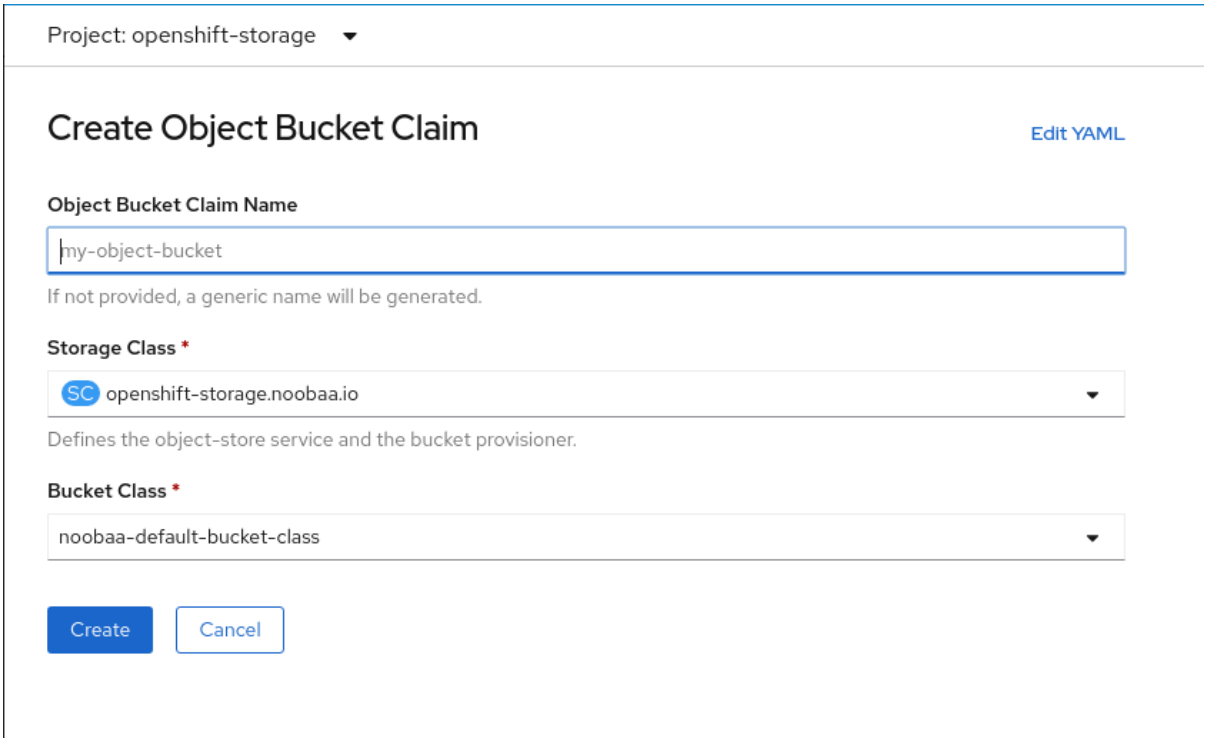
- Administrative access to the OpenShift Web Console.

Procedure

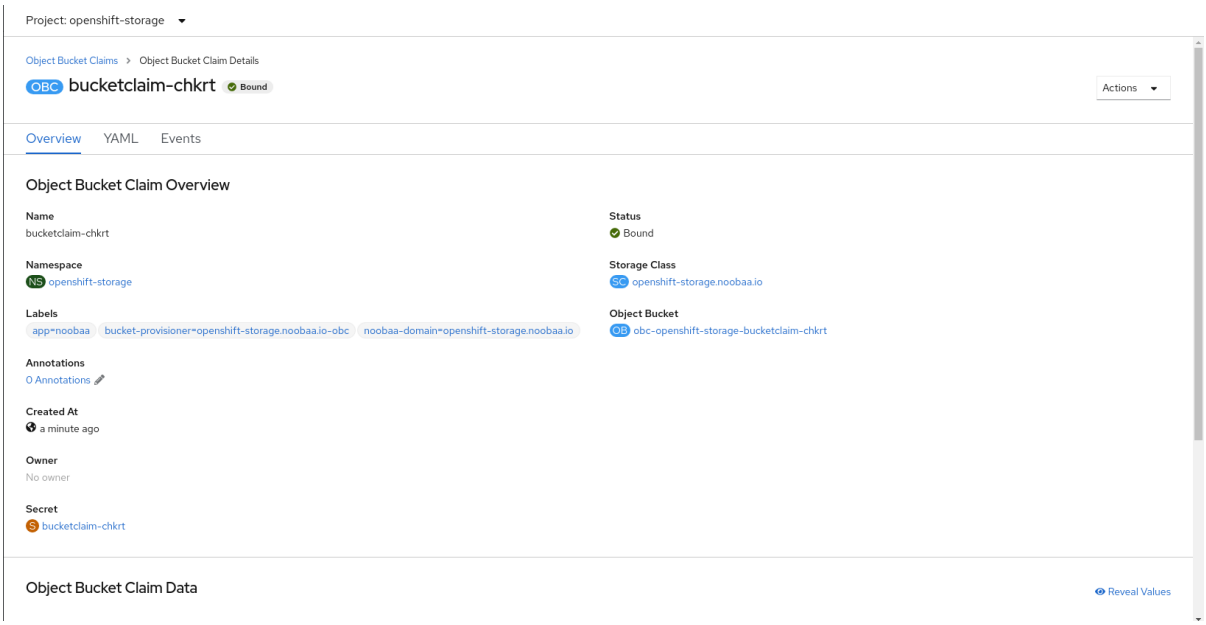
1. Log into the OpenShift Web Console.
2. On the left navigation bar, click **Storage → Object Bucket Claims**.
3. In the following window, click **Create Object Bucket Claim**:



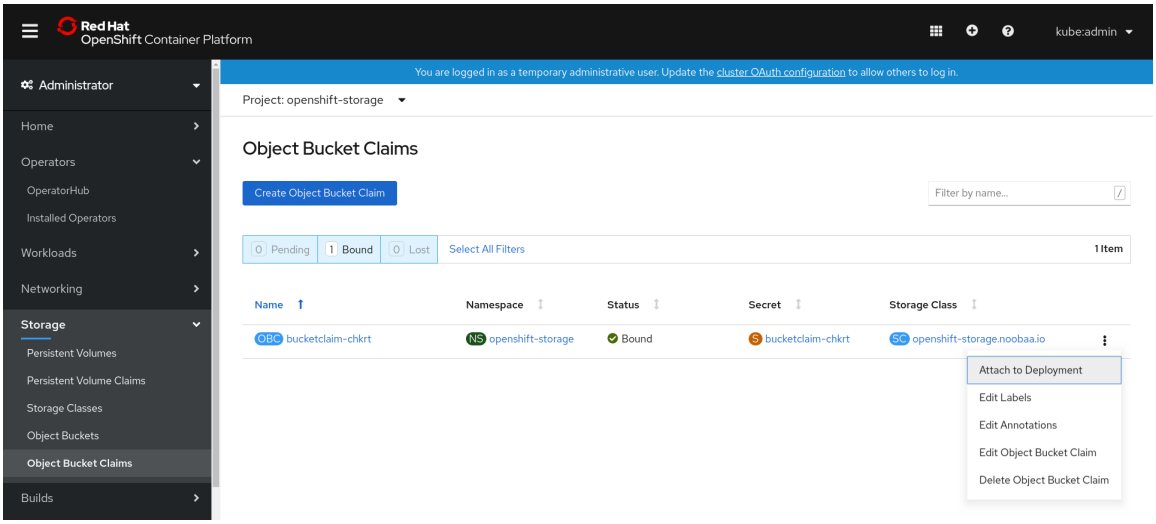
4. In the following window, enter a name for your object bucket claim, and select the appropriate storage class and bucket class from the dropdown menus:



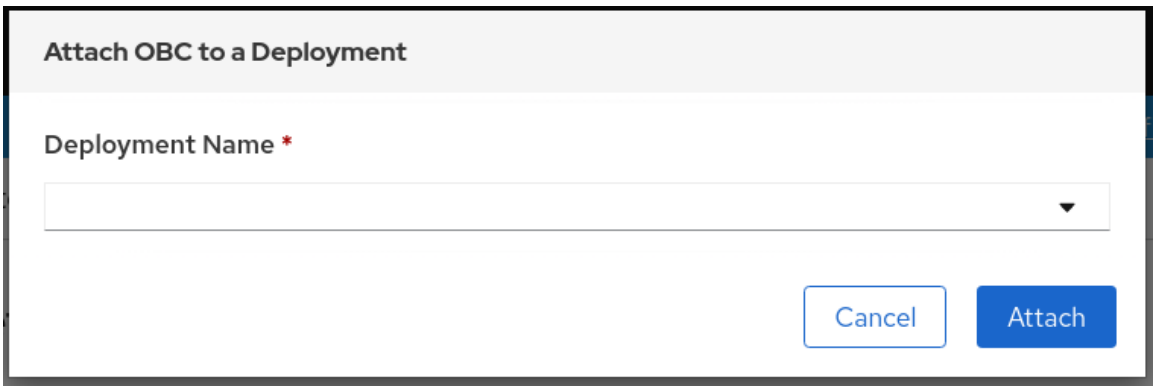
5. Click **Create**.
   Once the OBC is created, you will be redirected to its detail page:

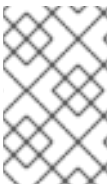


6. Once you've created the OBC, you can attach it to a deployment.
   a. On the left navigation bar, click **Storage → Object Bucket Claims**.
   b. Click the action menu (⋮) next to the OBC you created.
   c. From the drop down menu, select **Attach to Deployment**



d. In the following window, select the desired deployment from the drop down menu, then click **Attach**:



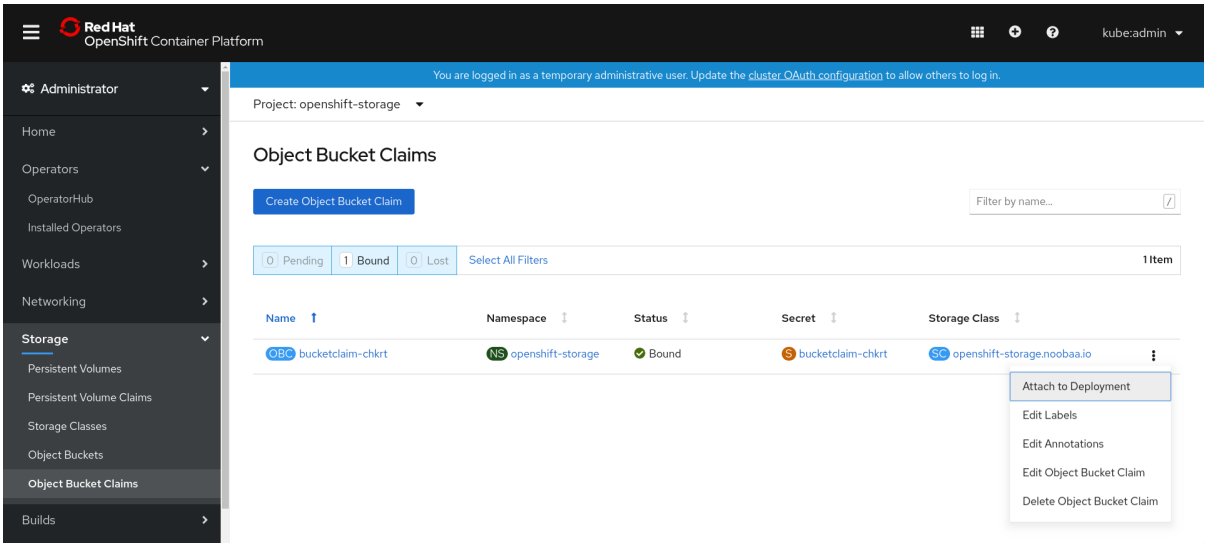


NOTE

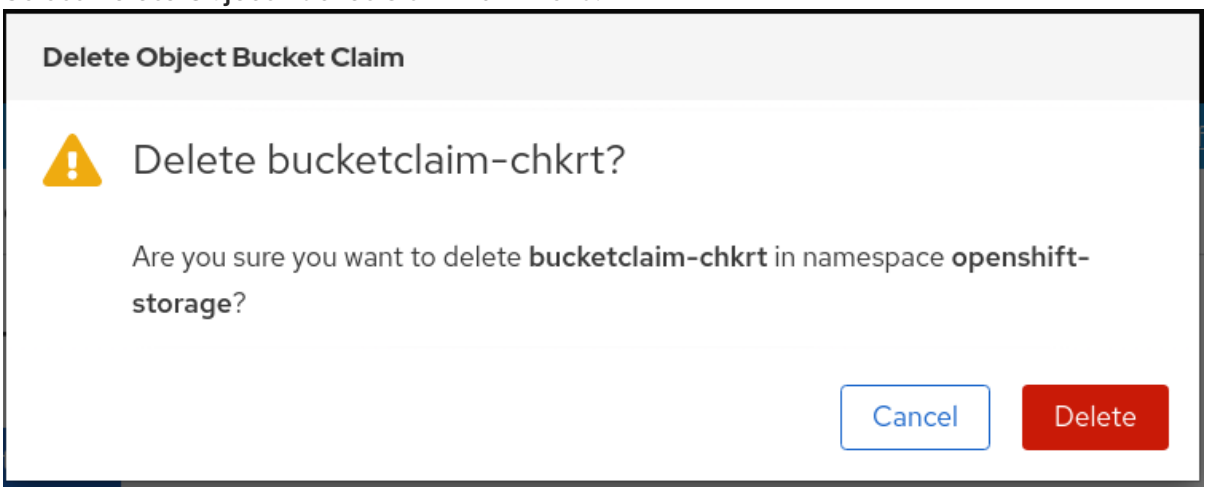
In order for your applications to communicate with the OBC, you need to use the configmap and secret. For more information about this, see Section 7.6.1, “Dynamic Object Bucket Claim”.

7.6.3.1. Delete an Object Bucket Claim

1. On the **Object Bucket Claims** page, click on the action menu (⋮) next to the OBC that you want to delete.



2. Select **Delete Object Bucket Claim** from menu.



3. Click **Delete**.

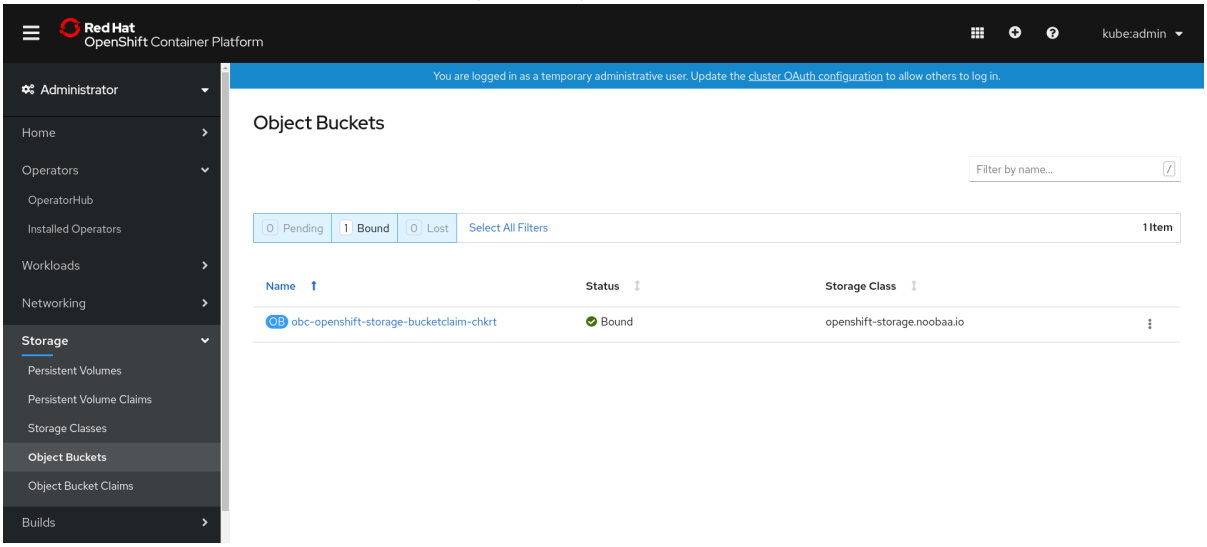
7.6.3.2. Viewing object buckets using the Multicloud Object Gateway user interface

You can view the details of object buckets created for Object Bucket Claims (OBCs).

Procedure

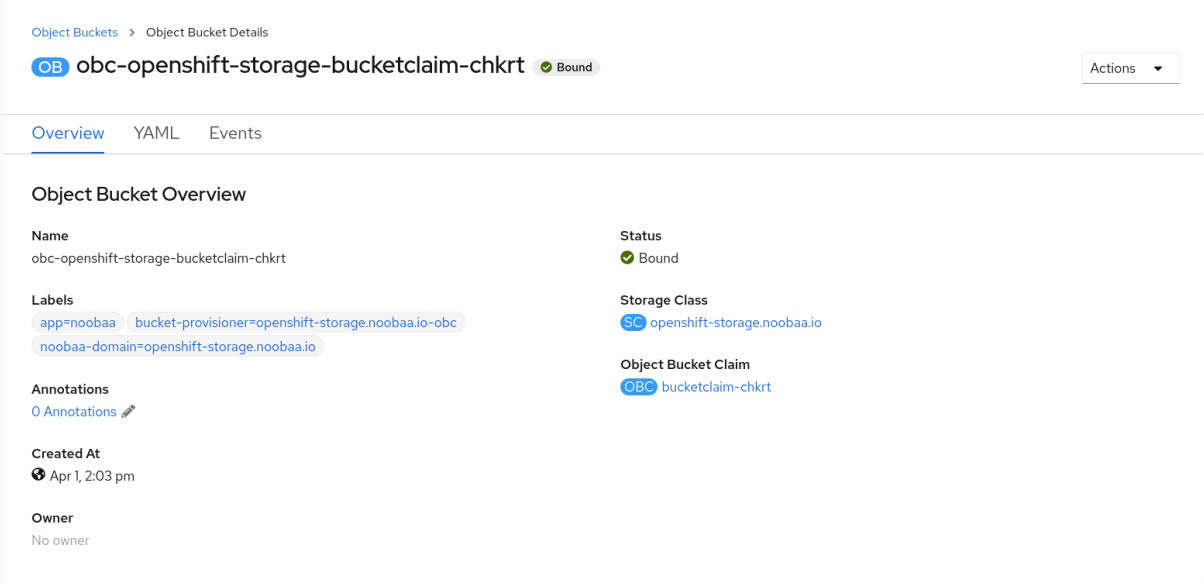
To view the object bucket details:

1. Log into the OpenShift Web Console.
2. On the left navigation bar, click **Storage → Object Buckets**:



   You can also navigate to the details page of a specific OBC and click the **Resource** link to view the object buckets for that OBC.

3. Select the object bucket you want to see details for. You will be navigated to the object bucket's details page:



7.7. 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

7.7.1. 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 data chunking, deduplication, compression, and encryption, and it accepts data placement instructions from the Multicloud Object Gateway.

7.7.2. Scaling with storage nodes

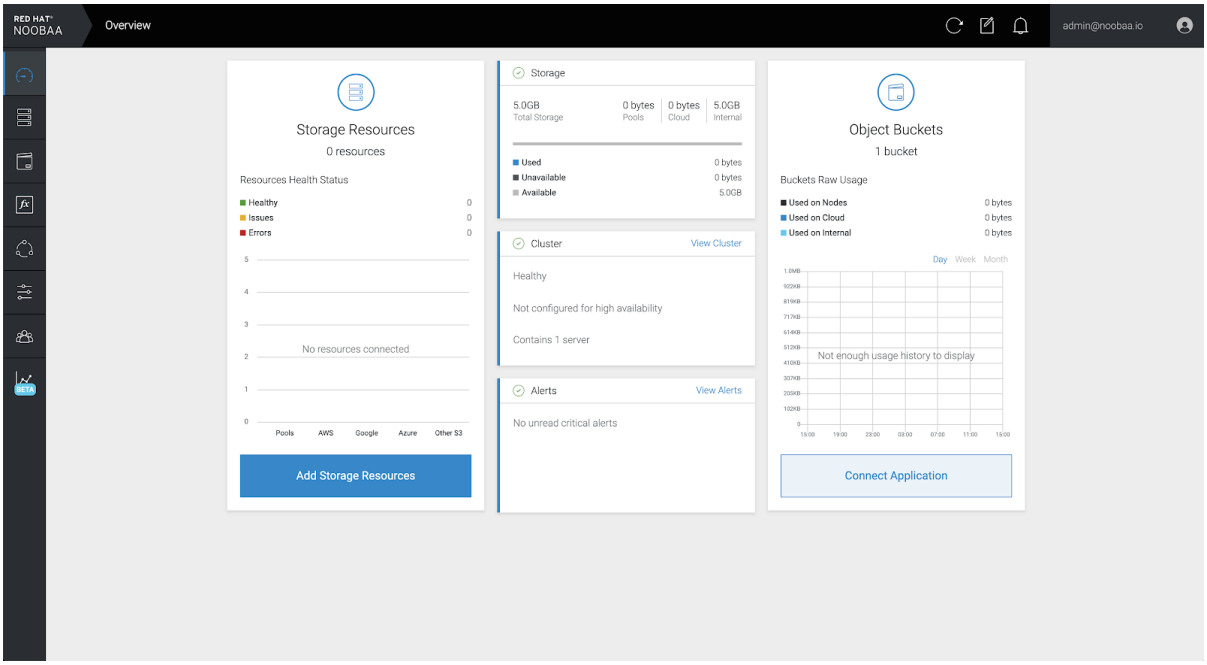
Prerequisites

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

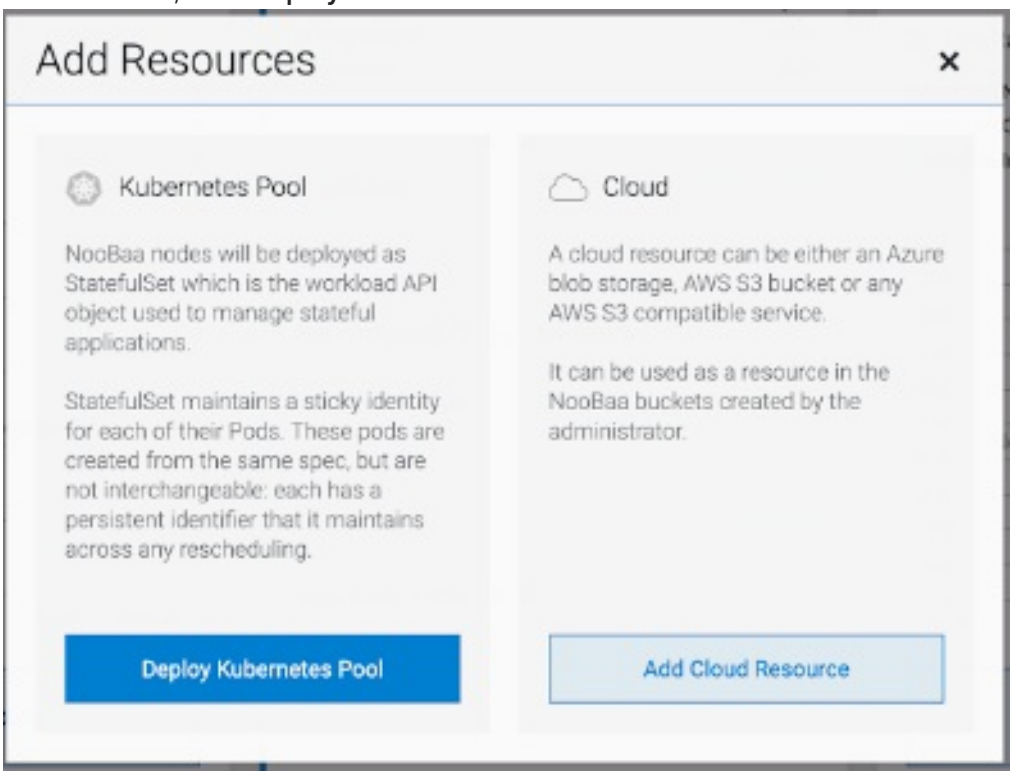
A storage node in the Multicloud Object Gateway is a NooBaa daemon container attached to one or more persistent volumes and used for local object service data storage. NooBaa daemons can be deployed on Kubernetes nodes. This can be done by creating a Kubernetes pool consisting of StatefulSet pods.

Procedure

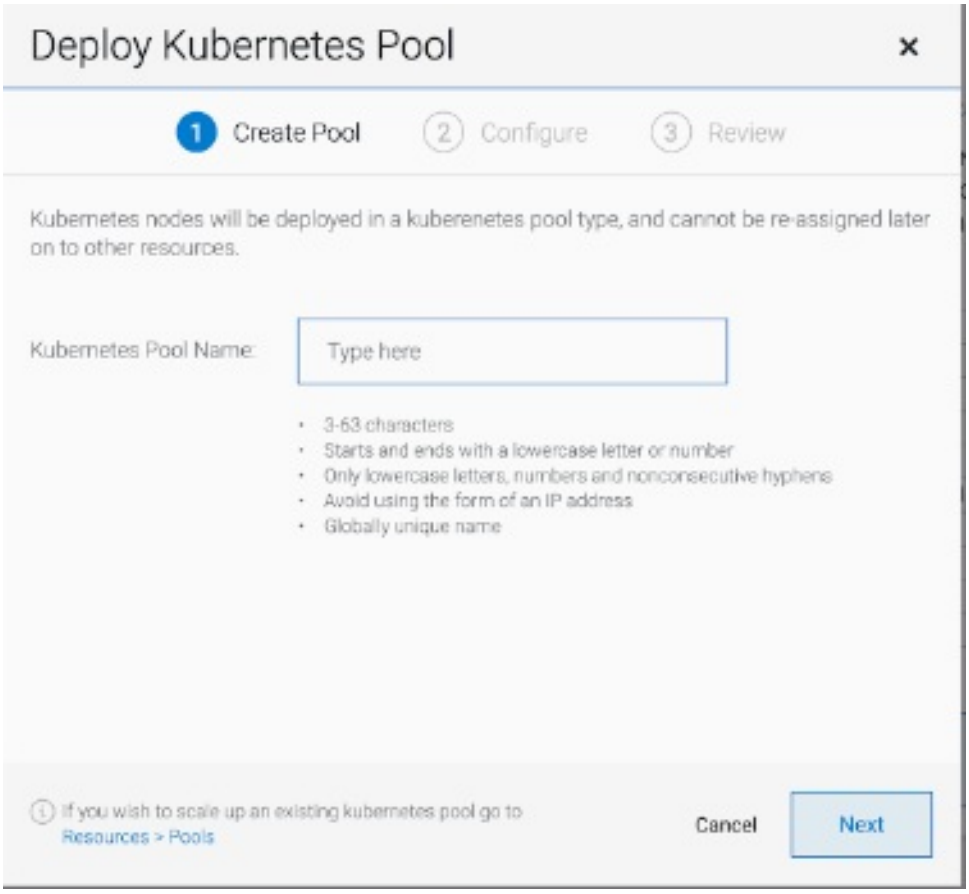
1. In the Mult-Cloud Object Gateway user interface, from the **Overview** page, click **Add Storage Resources**:



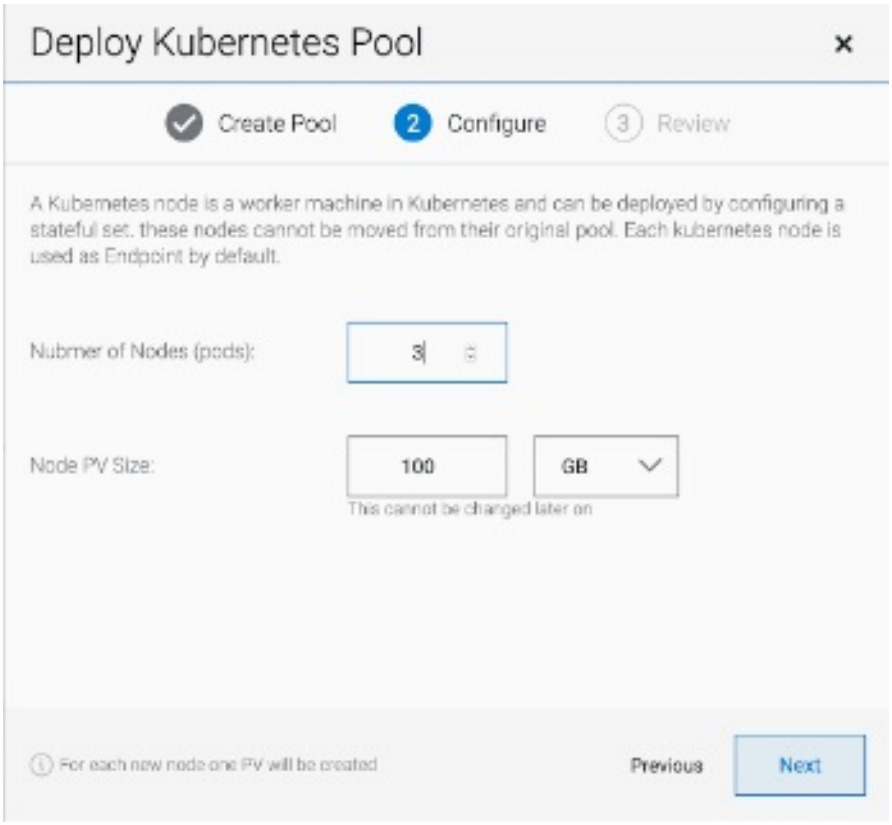
2. In the window, click **Deploy Kubernetes Pool**



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

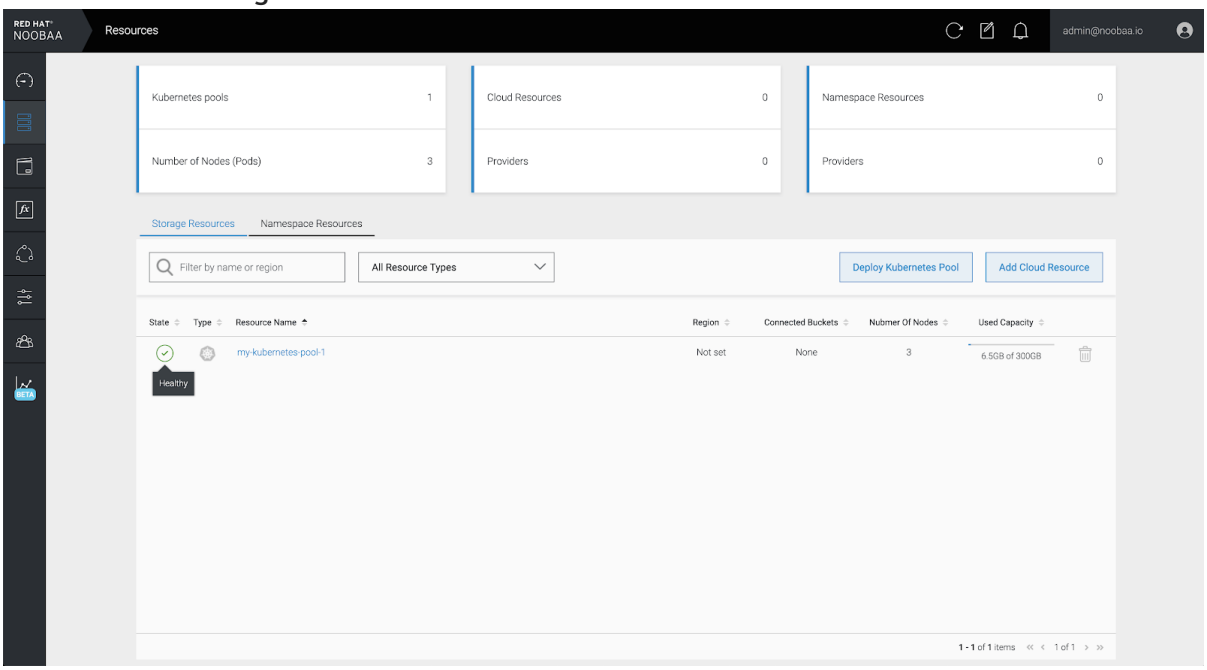


4. In the **Configure** step, configure the number of requested pods and the size of each PV. For each new pod, one PV is be created.



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 8. MANAGING PERSISTENT VOLUME CLAIMS

IMPORTANT

Expanding PVCs is not supported for PVCs backed by OpenShift Container Storage.

8.1. 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**.

8.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS

WARNING

Expanding Persistent Volume Claims (PVCs) is not supported for PVCs backed by OpenShift Container Storage.

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.
4. Check the **Status** column corresponding to the required PVC.
5. Click the required **Name** to view the PVC details.

8.3. 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.

8.4. DYNAMIC PROVISIONING

8.4.1. 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.

8.4.2. 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 4.4 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 4.4, 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**

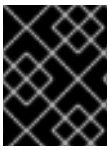
The judgement of which driver (RBD or CephFS) to use is based on the entry in the **storageclass.yaml** file.

8.4.3. 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
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.
Ceph File System (POSIX Compliant filesystem)	**openshift-storage.cephfs.csi.ceph.com**	Provisions a volume for ReadWriteMany (RWX) or ReadWriteOnce (RWO) access modes using the Ceph Filesytem configured in a Ceph cluster.
Ceph RBD (Block Device)	**openshift-storage.rbd.csi.ceph.com**	Provisions a volume for RWO access mode for Ceph RBD, RWO and RWX access mode for block PVC, and RWO access mode for Filesystem PVC.
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.

Storage type	Provisioner plug-in name	Notes
S3 Bucket (MCG Object Bucket Claim)	**openshift-storage.noobaa.io/obc**	Provisions an object bucket claim to support S3 API calls through the Multicloud Object Gateway (MCG). The exact storage backing the S3 bucket is dependent on the MCG configuration and the type of deployment.
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 9. REPLACING STORAGE NODES

You can choose one of the following procedures to replace storage nodes:

- Section 9.1, “Replacing operational nodes on Azure installer-provisioned infrastructure”
- Section 9.2, “Replacing failed nodes on Azure installer-provisioned infrastructure”

9.1. REPLACING OPERATIONAL NODES ON AZURE INSTALLER-PROVISIONED INFRASTRUCTURE

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

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. Mark the node as unschedulable using the following command:

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

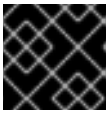
4. Drain the node using the following command:

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

IMPORTANT

This activity may take at least 5-10 minutes or more. Ceph errors generated during this period are temporary and are automatically resolved when the new node is labeled and functional.

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



IMPORTANT

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

9. Click **Compute → Nodes**, confirm if the new node is in **Ready** state.
10. 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=""
```

11. Restart the **mgr** pod to update the OpenShift Container Storage with the new hostname.

```
$ oc delete pod rook-ceph-mgr-xxxx
```

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.
4. If verification steps fail, kindly contact Red Hat Support .

9.2. REPLACING FAILED NODES ON AZURE INSTALLER-PROVISIONED INFRASTRUCTURE

Perform this procedure to replace a failed node which is not operational on Azure installer-provisioned infrastructure (IPI) for OpenShift Container Storage.

Procedure

1. Log in to OpenShift Web Console and click **Compute → Nodes**.
2. Identify the faulty node and click on its **Machine Name**.
3. Click **Actions → Edit Annotations**, and click **Add More**.
4. Add **machine.openshift.io/exclude-node-draining** and click **Save**.
5. Click **Actions → Delete Machine**, and click **Delete**.
6. A new machine is automatically created, wait for new machine to start.

IMPORTANT

This activity may take at least 5-10 minutes or more. Ceph errors generated during this period are temporary and are automatically resolved when the new node is labeled and functional.

7. Click **Compute → Nodes**, confirm if the new node is in **Ready** state.
8. 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=""
     ```

9. [Optional]: If the failed Azure instance is not removed automatically, terminate the instance from Azure console.

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.
4. If verification steps fail, kindly contact Red Hat Support.

CHAPTER 10. REPLACING STORAGE DEVICES

10.1. REPLACING OPERATIONAL OR FAILED STORAGE DEVICES ON AZURE INSTALLER-PROVISIONED INFRASTRUCTURE

When you need to replace a device in a dynamically created storage cluster on an Azure installer-provisioned infrastructure, you must replace the storage node. For information about how to replace nodes, see:

- Replacing operational nodes on Azure installer-provisioned infrastructure
- Replacing failed nodes on Azure installer-provisioned infrastructures.

CHAPTER 11. UPDATING OPENSHIFT CONTAINER STORAGE

It is recommended to use the same version of Red Hat OpenShift Container Platform with Red Hat OpenShift Container Storage. Upgrade Red Hat OpenShift Container Platform first, and then upgrade Red Hat OpenShift Container Storage. Refer to this Red Hat Knowledgebase article for a complete OpenShift Container Platform and OpenShift Container Storage supportability and compatibility matrix.

If using Local Storage Operator, Local Storage Operator version must match with the Red Hat OpenShift Container Platform version in order to have the Local Storage Operator fully supported with Red Hat OpenShift Container Storage. Local Storage Operator does not get upgraded when Red Hat OpenShift Container Platform is upgraded. To check if your OpenShift Container Storage cluster uses the Local Storage Operator, see the Checking for Local Storage Operator deployments section of the Troubleshooting Guide.

IMPORTANT

If your cluster was deployed using local storage devices and uses the Local Storage Operator in Openshift Container Storage version 4.3, you must re-install the cluster and not upgrade to version 4.4. For details on installation, see Installing OpenShift Container Storage using local storage devices.

11.1. ENABLING AUTOMATIC UPDATES FOR OPENSHIFT CONTAINER STORAGE OPERATOR

Use this procedure to enable automatic update approval for updating OpenShift Container Storage operator in OpenShift Container Platform.

Prerequisites

- Update the OpenShift Container Platform cluster to the latest stable release of version 4.3.X or 4.4.Y, see Updating Clusters.
- Switch the Red Hat OpenShift Container Storage channel channel from **stable-4.3** to **stable-4.4**. For details about channels, see OpenShift Container Platform upgrade channels and releases.

NOTE

You are required to switch channels only when you are updating minor versions (for example, updating from 4.3 to 4.4) and not when updating between batch updates of 4.4 (for example, updating from 4.4.0 to 4.4.1).

- Ensure that all OpenShift Container Storage nodes are in **Ready** status.
- Under **Persistent Storage** in **Status** card, confirm that the Ceph cluster is healthy and data is resilient.
- Ensure that you have sufficient time to complete the Openshift Container Storage (OCS) 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 on the OpenShift Container Storage operator name.
5. Click **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**.
     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 **Status** card confirm that the OpenShift Container Storage cluster has a green tick mark indicating it is healthy.
2. Click **Operators → Installed Operators → OpenShift Container Storage Operator**.
3. Under **Storage Cluster**, verify that the cluster service status in **Ready**.

NOTE

Once updated from OpenShift Container Storage version 4.3 to 4.4, the **Version** field here will still display 4.3. This is because the **ocs-operator** does not update the string represented in this field.

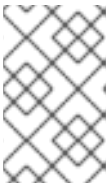
4. If verification steps fail, kindly contact Red Hat Support .

11.2. MANUALLY UPDATING OPENSHIFT CONTAINER STORAGE OPERATOR

Use this procedure to update OpenShift Container Storage operator by providing manual approval to the install plan.

Prerequisites

- Update the OpenShift Container Platform cluster to the latest stable release of version 4.3.X or 4.4.Y, see Updating Clusters.
- Switch the Red Hat OpenShift Container Storage channel channel from **stable-4.3** to **stable-4.4**. For details about channels, see OpenShift Container Platform upgrade channels and releases.



NOTE

You are required to switch channels only when you are updating minor versions (for example, updating from 4.3 to 4.4) and not when updating between batch updates of 4.4 (for example, updating from 4.4.0 to 4.4.1).

- Ensure that all OpenShift Container Storage nodes are in **Ready** status.
- Under **Persistent Storage** in **Status** card, confirm that the Ceph cluster is healthy and data is resilient.
- Ensure that you have sufficient time to complete the Openshift Container Storage (OCS) 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 **Subscription** tab and click the link under **Approval**.
5. Select **Manual** and click **Save**.
6. Wait for the **Upgrade Status** to change to **Upgrading**.
7. If the **Upgrade Status** shows **requires approval**, click on **requires approval**.
8. On the **InstallPlan Details** page, click **Preview Install Plan**.
9. Review the install plan and click **Approve**.
10. Wait for the **Status** to change from **Unknown** to **Created**.
11. Click **Operators → Installed Operators**
12. Select the **openshift-storage** project.

13. Wait for the **Status** to change to **Up to date**

Verification steps

1. Click **Overview → Persistent Storage** tab and in **Status** card confirm that the Ceph cluster has a green tick mark indicating it is healthy.
2. Click **Operators → Installed Operators → OpenShift Container Storage Operator**.
3. Under **Storage Cluster**, verify that the cluster service status in **Ready**.

NOTE

Once updated from OpenShift Container Storage version 4.3 to 4.4, the **Version** field here will still display 4.3. This is because the **ocs-operator** does not update the string represented in this field.

4. If verification steps fail, kindly contact Red Hat Support .