Red Hat OpenShift Container Storage 4.5

Deploying and managing OpenShift Container Storage using Red Hat Virtualization platform

How to install and manage

Last Updated: 2021-03-12

Red Hat OpenShift Container Storage 4.5 Deploying and managing OpenShift Container Storage using Red Hat Virtualization platform

How to install and manage

Abstract

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

Table of Contents

PREFACE 4

CHAPTER 1. DEPLOYING OPENSHIFT CONTAINER STORAGE ON RED HAT VIRTUALIZATION PLATFORM 5
1.1. REQUIREMENTS FOR INSTALLING OPENSHIFT CONTAINER STORAGE USING LOCAL STORAGE DEVICES 5
1.2. INSTALLING RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR 6
1.3. INSTALLING LOCAL STORAGE OPERATOR 8
1.4. FINDING AVAILABLE STORAGE DEVICES 9
1.5. CREATING OPENSHIFT CONTAINER STORAGE CLUSTER ON RED HAT VIRTUALIZATION PLATFORM 10

CHAPTER 2. VERIFYING OPENSHIFT CONTAINER STORAGE DEPLOYMENT 16
2.1. VERIFYING THE STATE OF THE PODS 16
2.2. VERIFYING THE OPENSHIFT CONTAINER STORAGE CLUSTER IS HEALTHY 18
2.3. VERIFYING THE MULTICLOUD OBJECT GATEWAY IS HEALTHY 19
2.4. VERIFYING THAT THE OPENSHIFT CONTAINER STORAGE SPECIFIC STORAGE CLASSES EXIST 20

CHAPTER 3. UNINSTALLING OPENSHIFT CONTAINER STORAGE 21
3.1. UNINSTALLING OPENSHIFT CONTAINER STORAGE ON INTERNAL MODE 21
3.2. REMOVING MONITORING STACK FROM OPENSHIFT CONTAINER STORAGE 24
3.3. REMOVING OPENSHIFT CONTAINER PLATFORM REGISTRY FROM OPENSHIFT CONTAINER STORAGE 27
3.4. REMOVING THE CLUSTER LOGGING OPERATOR FROM OPENSHIFT CONTAINER STORAGE 28

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. REQUIREMENTS FOR SCALING STORAGE NODES 41
6.2. SCALING UP STORAGE BY ADDING CAPACITY TO YOUR OPENSHIFT CONTAINER STORAGE NODES USING LOCAL STORAGE DEVICES 41
6.3. SCALING OUT STORAGE CAPACITY BY ADDING NEW NODES 45
6.3.1. Adding a node using a local storage device 45
6.3.2. Verifying the addition of a new node 46
6.3.3. Scaling up storage capacity 46

CHAPTER 7. MULTICLOUD OBJECT GATEWAY 47
7.1. ABOUT THE MULTICLOUD OBJECT GATEWAY 47
7.2. ACCESSING THE MULTICLOUD OBJECT GATEWAY WITH YOUR APPLICATIONS 47
7.2.1. Accessing the Multicloud Object Gateway from the terminal 47
7.2.2. Accessing the Multicloud Object Gateway from the MCG command-line interface 49
7.3. ADDING STORAGE RESOURCES FOR HYBRID OR MULTICLOUD 52
7.3.1. Adding storage resources for hybrid or Multicloud using the MCG command line interface 52
7.3.2. Creating an S3 compatible NooBaa backingstore 53
7.3.3. Adding storage resources for hybrid and Multicloud using the user interface 55
7.3.4. Creating a new bucket class 57
7.3.5. Creating a new backing store 59

7.4. MIRRORING DATA FOR HYBRID AND MULTICLOUD BUCKETS 61
7.4.1. Creating bucket classes to mirror data using the MCG command-line-interface 61
7.4.2. Creating bucket classes to mirror data using a YAML 61
7.4.3. Configuring buckets to mirror data using the user interface 62
7.5. BUCKET POLICIES IN THE MULTICLOUD OBJECT GATEWAY 64
7.5.1. About bucket policies 64
7.5.2. Using bucket policies 64
7.5.3. Creating an AWS S3 user in the Multicloud Object Gateway 66
7.6. OBJECT BUCKET CLAIM 68
7.6.1. Dynamic Object Bucket Claim 68
7.6.2. Creating an Object Bucket Claim using the command line interface 70
7.6.3. Creating an Object Bucket Claim using the OpenShift Web Console 73
7.7. SCALING MULTICLOUD OBJECT GATEWAY PERFORMANCE BY ADDING ENDPOINTS 74
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 78
8.1. CONFIGURING APPLICATION PODS TO USE OPENSHIFT CONTAINER STORAGE 78
8.2. VIEWING PERSISTENT VOLUME CLAIM REQUEST STATUS 79
8.3. REVIEWING PERSISTENT VOLUME CLAIM REQUEST EVENTS 80
8.4. EXPANDING PERSISTENT VOLUME CLAIMS 80
8.5. DYNAMIC PROVISIONING 82
8.5.1. About dynamic provisioning 82
8.5.2. Dynamic provisioning in OpenShift Container Storage 82
8.5.3. Available dynamic provisioning plug-ins 83

CHAPTER 9. REPLACING FAILED STORAGE NODES ON RED HAT VIRTUALIZATION PLATFORM 85

CHAPTER 10. REPLACING FAILED STORAGE DEVICES ON RED HAT VIRTUALIZATION PLATFORM . . . 92

PREFACE

Red Hat OpenShift Container Storage 4.5 supports deployment on existing Red Hat OpenShift Container Platform (OCP) Red Hat Virtualization platform clusters.

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

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

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

NOTE

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

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

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

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

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

- You must have at least three OpenShift Container Platform worker nodes in the cluster with locally attached storage devices on each of them.
    - Each of the three selected nodes must have at least one raw block device available to be used by OpenShift Container Storage.
    - For minimum starting node requirements, see Resource requirements section in Planning guide.
    - The devices to be used must be empty, that is, there should be no PVs, VGs, or LVs remaining on the disks.
- You must have a minimum of three labeled nodes.
    - Each node that has local storage devices to be used by OpenShift Container Storage must have a specific label to deploy OpenShift Container Storage pods. To label the nodes, use the following command:

```
$ oc label nodes <NodeNames> cluster.ocs.openshift.io/openshift-storage=''
```

- There should not be any storage providers managing locally mounted storage on the storage nodes that would conflict with the use of Local Storage Operator for Red Hat OpenShift Container Storage.

- The Local Storage Operator version must match the Red Hat OpenShift Container Platform version in order to have the Local Storage Operator fully supported with Red Hat OpenShift Container Storage. The Local Storage Operator does not get upgraded when Red Hat OpenShift Container Platform is upgraded.

1.2. INSTALLING RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR

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

Prerequisites

- You must be logged into the OpenShift Container Platform cluster.

- You must have at least three worker nodes in the OpenShift Container Platform cluster.

NOTE

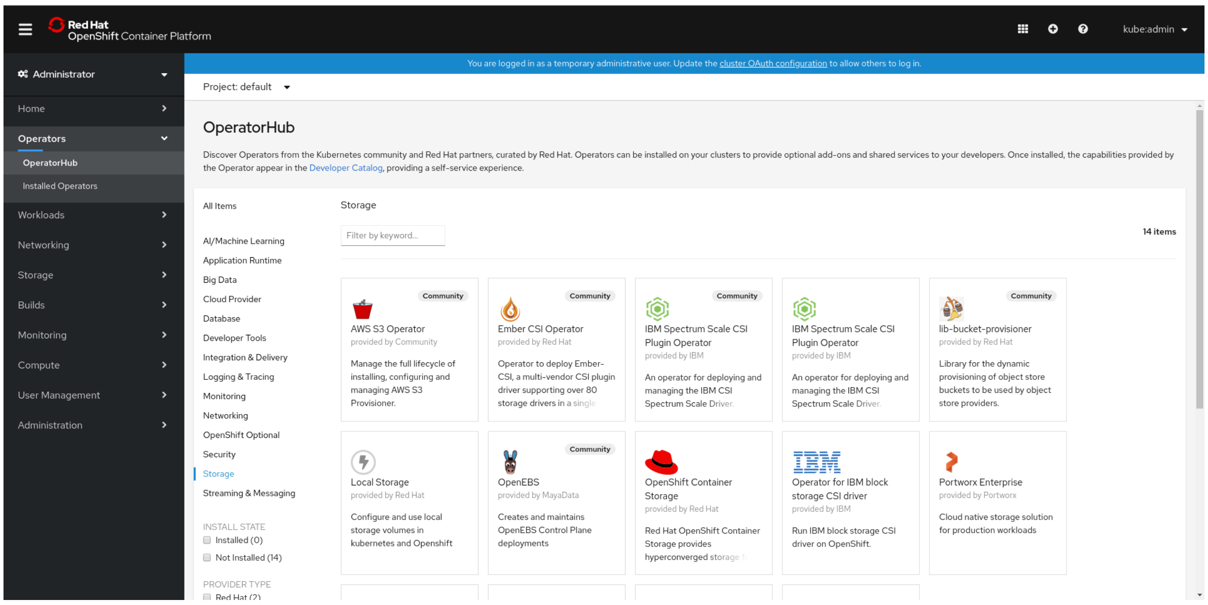
When you need to override the cluster-wide default node selector for OpenShift Container Storage, you can use the following command in command line interface to specify a blank node selector for the **openshift-storage** namespace:

```
$ oc annotate namespace openshift-storage openshift.io/node-selector=
```

Procedure

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

Figure 1.1. List of operators in the Operator Hub



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: default

OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. Operators can be installed on your clusters to provide optional add-ons and shared services to your developers. Once installed, the capabilities provided by the Operator appear in the Developer Catalog, providing a self-service experience.

Community
AWS S3 Operator
provided by Community
Manage the full lifecycle of installing, configuring and managing AWS S3 Provisioner.

Community
Ember CSI Operator
provided by Red Hat
Operator to deploy Ember-CSI, a multi-vendor CSI plugin driver supporting over 80 storage drivers in a single

Community
IBM Spectrum Scale CSI Plugin Operator
provided by IBM
An operator for deploying and managing the IBM CSI Spectrum Scale Driver.

IBM Spectrum Scale CSI Plugin Operator
provided by IBM
An operator for deploying and managing the IBM CSI Spectrum Scale Driver.

Community
lib-bucket-provisioner
provided by Red Hat
Library for the dynamic provisioning of object store buckets to be used by object store providers.

Local Storage
provided by Red Hat
Configure and use local storage volumes in kubernetes and Openshift

Community
OpenEBS
provided by MayaData
Creates and maintains OpenEBS Control Plane deployments

OpenShift Container Storage
provided by Red Hat
Red Hat OpenShift Container Storage provides hyperconverged storage

Operator for IBM block storage CSI driver
provided by IBM
Run IBM block storage CSI driver on OpenShift.

Portworx Enterprise
provided by Portworx
Cloud native storage solution for production workloads

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-4.5**

   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 shows the Status as **Succeeded** on the Installed Operators dashboard.

1.3. INSTALLING LOCAL STORAGE OPERATOR

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

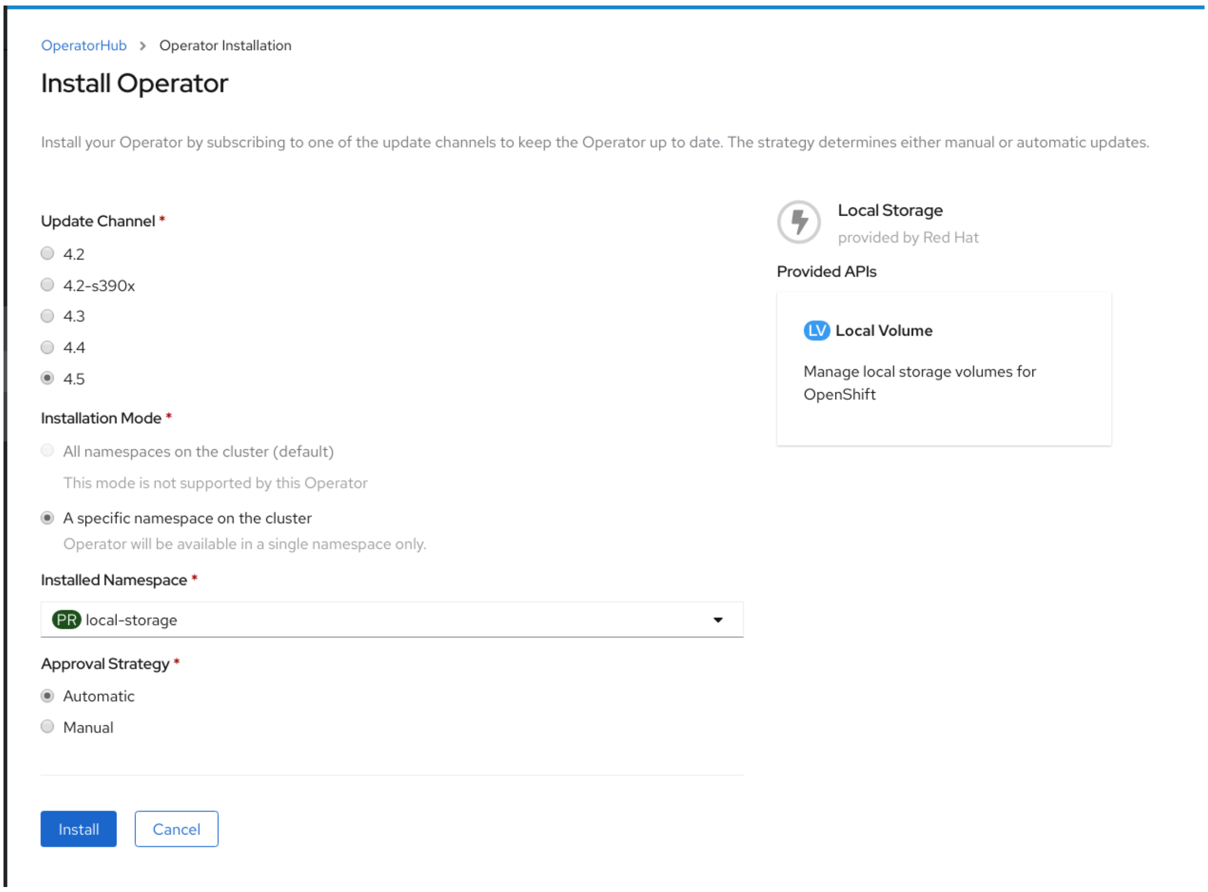
Prerequisites

- Create a namespace called **local-storage** as follows:
  a. Click **Administration → Namespaces** in the left pane of the OpenShift Web Console.
  b. Click **Create Namespace**.
  c. In the Create Namespace dialog box, enter **local-storage** for Name.
  d. Select **No restrictions** option for **Default Network Policy**.
  e. Click **Create**.

Procedure

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

Figure 1.2. Install Operator page



OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update Channel *

- 4.2
- 4.2-s390x
- 4.3
- 4.4
- [✓ 4.5]

Installation Mode *

- All namespaces on the cluster (default)
  This mode is not supported by this Operator
- [✓ A specific namespace on the cluster]
  Operator will be available in a single namespace only.

Installed Namespace *

PR local-storage

Approval Strategy *

- [✓ Automatic]
- Manual

Local Storage
provided by Red Hat

Provided APIs

LV Local Volume

Manage local storage volumes for OpenShift

4. On the **Install Operator** page, ensure the following options are selected
   a. Update Channel as **stable-4.5**
   b. Installation Mode as **A specific namespace on the cluster**
   c. Installed Namespace as **local-storage**.
   d. Approval Strategy as **Automatic**
5. Click **Install**.
6. Verify that the Local Storage Operator shows the Status as **Succeeded**.

1.4. FINDING AVAILABLE STORAGE DEVICES

Use this procedure to identify the device names for each of the three or more worker nodes that you have labeled with the OpenShift Container Storage label **cluster.ocs.openshift.io/openshift-storage=''** before creating PVs for Red Hat Virtualization.

Procedure

1. List and verify the name of the worker nodes with the OpenShift Container Storage label.

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

Example output:

```
NAME         STATUS   ROLES    AGE     VERSION
rhvworker01    Ready    worker   6h45m   v1.16.2
rhvworker02    Ready    worker   6h45m   v1.16.2
rhvworker03    Ready    worker   6h45m   v1.16.2
```

2. Log in to each worker node that is used for OpenShift Container Storage resources and find the unique **by-id** device name for each available raw block device.

   ```
   $ oc debug node/<Nodename>
   ```

   Example output:

   ```
   $ oc debug node/rhvworker01
   Starting pod/rhvworker01-debug ...
   To use host binaries, run `chroot /host`
   Pod IP: 10.0.135.71
   If you don't see a command prompt, try pressing enter.
   sh-4.2# chroot /host
   sh-4.4# lsblk
   NAME                         MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
   xvda                        202:0    0   120G  0 disk
   |-xvda1                     202:1    0   384M  0 part /boot
   |-xvda2                     202:2    0   127M  0 part /boot/efi
   |-xvda3                     202:3    0     1M  0 part
   `-xvda4                     202:4    0 119.5G  0 part
     `-coreos-luks-root-nocrypt 253:0    0 119.5G  0 dm   /sysroot
   nvme0n1                     259:0    0   931G  0 disk
   ```

   In this example, for **rhvworker01**, the available local device is **nvme0n1**.

3. Identify the unique ID for each of the devices selected in Step 2.

   ```
   sh-4.4#  ls -l /dev/disk/by-id/ | grep nvme0n1
   lrwxrwxrwx. 1 root root 13 Mar 17 16:24 nvme-
   INTEL_SSDPE2KX010T7_PHLF733402LM1P0GGN -> ../../nvme0n1
   ```

   In the above example, the ID for the local device **nvme0n1**

   ```
   nvme-INTEL_SSDPE2KX010T7_PHLF733402LM1P0GGN
   ```

4. Repeat the above step to identify the device ID for all the other nodes that have the storage devices to be used by OpenShift Container Storage. See this Knowledge Base article for more details.

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

Prerequisites

- Ensure that all the requirements in the Requirements for installing OpenShift Container Storage using local storage devices section are met.

- Verify your OpenShift Container Platform worker nodes are labeled for OpenShift Container Storage:

```
$ oc get nodes -l cluster.ocs.openshift.io/openshift-storage -o jsonpath='{range .items[*]}
{.metadata.name}{"\n"}'
```

To identify storage devices on each node, refer to Finding available storage devices.

Procedure

1. Create the **LocalVolume** CR for block PVs.
   Example of **LocalVolume** CR **local-storage-block.yaml** using OCS label as node selector.

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: local-block
  namespace: local-storage
  labels:
    app: ocs-storagecluster
spec:
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
        - key: cluster.ocs.openshift.io/openshift-storage
          operator: In
          values:
          - ""
  storageClassDevices:
    - storageClassName: localblock
      volumeMode: Block
      devicePaths:
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY81260978128A   # <-- modify
this line
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY80440W5U128A   # <-- modify
this line
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYB85AABDE128A   # <-- modify
this line
```

2. Create the **LocalVolume** CR for block PVs.

```
$ oc create -f local-storage-block.yaml
```

3. Check if the pods are created.
   Example output:

```
NAME                                      READY   STATUS  RESTARTS AGE
local-block-local-diskmaker-cmfql         1/1     Running 0        31s
local-block-local-diskmaker-g6fzr         1/1     Running 0        31s
local-block-local-diskmaker-jkqxt         1/1     Running 0        31s
local-block-local-provisioner-jgqcc       1/1     Running 0        31s
local-block-local-provisioner-mx49d       1/1     Running 0        31s
local-block-local-provisioner-qbcvp       1/1     Running 0        31s
local-storage-operator-54bc7566c6-ddbrt 1/1     Running 0        12m
```

4. Check if the PVs are created.

```
$ oc get pv
```

Example output:

```
NAME              CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
local-pv-150fdc87   931Gi      RWO            Delete           Available           localblock
2m11s
local-pv-183bfc0a   931Gi      RWO            Delete           Available           localblock
2m15s
local-pv-b2f5cb25   931Gi      RWO            Delete           Available           localblock
2m21s
```

5. Check for the new **localblock StorageClass**.

```
$ oc get sc | grep localblock
```

Example output:

```
NAME          PROVISIONER                    RECLAIMPOLICY
VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION     AGE
localblock    kubernetes.io/no-provisioner   Delete
WaitForFirstConsumer  false                2m10s
```

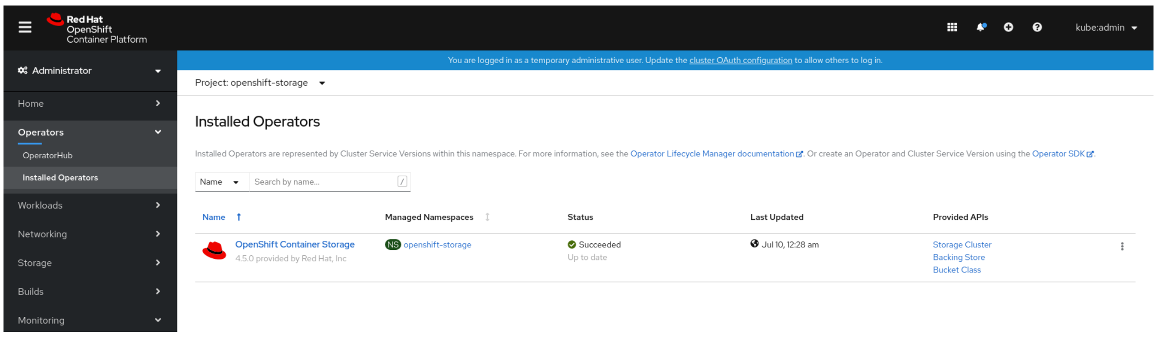
6. Create the OpenShift Container Storage Cluster Service that uses the **localblock** Storage Class.

   a. Log into the OpenShift Web Console.

   b. Click **Operators → Installed Operators** from the OpenShift Web Console to view the installed operators. Ensure that the **Project** selected is **openshift-storage**.

   c. On the **Installed Operators** page, click **Openshift Container Storage**.

   Figure 1.3. OpenShift Container Storage Operator page



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Installed Operators

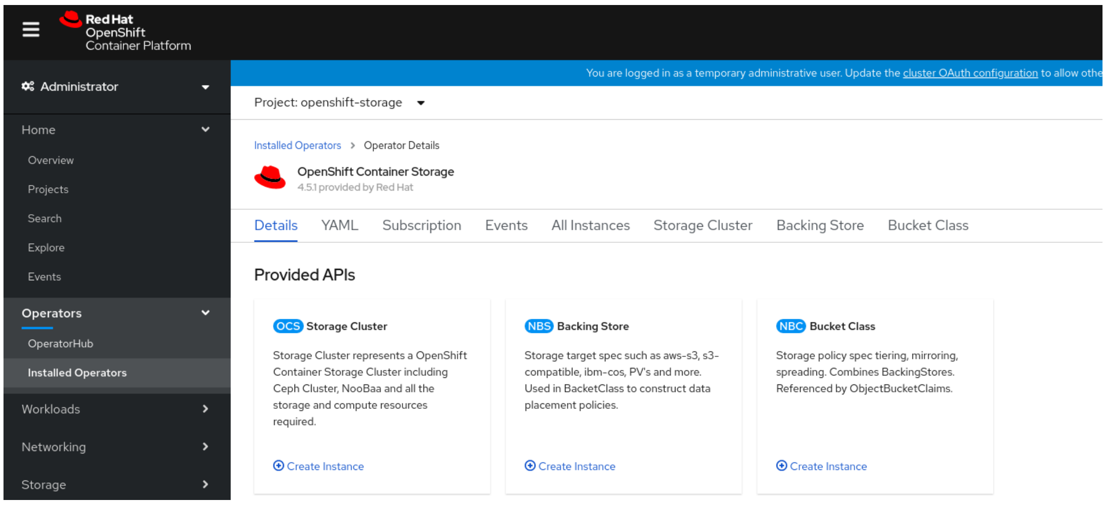
Installed Operators are represented by Cluster Service Versions within this namespace. For more information, see the Operator Lifecycle Manager documentation. Or create an Operator and Cluster Service Version using the Operator SDK.

Name	Managed Namespaces	Status	Last Updated	Provided APIs
OpenShift Container Storage 4.5.0 provided by Red Hat, Inc	NS openshift-storage	Succeeded Up to date	Jul 10, 12:28 am	Storage Cluster Backing Store Bucket Class

   d. On the **Installed Operators → Operator Details** page, perform either of the following to create a Storage Cluster Service.

      - On the **Details tab → Provided APIs → OCS Storage Cluster**, click **Create Instance**.

Figure 1.4. Operator Details Page



Red Hat OpenShift Container Platform

Administrator

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow othe

Project: openshift-storage

Home
- Overview
- Projects
- Search
- Explore
- Events

Operators
- OperatorHub
- Installed Operators

Workloads

Networking

Storage

Installed Operators > Operator Details

OpenShift Container Storage
4.5.1 provided by Red Hat

Details | YAML | Subscription | Events | All Instances | Storage Cluster | Backing Store | Bucket Class

Provided APIs

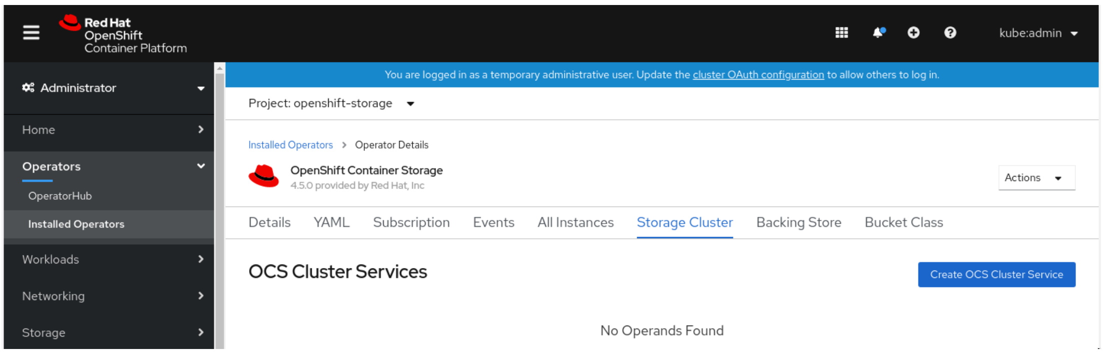
OCS Storage Cluster
Storage Cluster represents a OpenShift Container Storage Cluster including Ceph Cluster, NooBaa and all the storage and compute resources required.
Create Instance

NBS Backing Store
Storage target spec such as aws-s3, s3-compatible, ibm-cos, PV's and more. Used in BucketClass to construct data placement policies.
Create Instance

NBC Bucket Class
Storage policy spec tiering, mirroring, spreading. Combines BackingStores. Referenced by ObjectBucketClaims.
Create Instance

- Alternatively, select the **Storage cluster** tab and click **Create OCS Cluster Service**.

Figure 1.5. Storage Cluster tab



Red Hat OpenShift Container Platform

kube:admin

Administrator

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: openshift-storage

Home

Operators
- OperatorHub
- Installed Operators

Workloads

Networking

Storage

Installed Operators > Operator Details

OpenShift Container Storage
4.5.0 provided by Red Hat, Inc

Actions

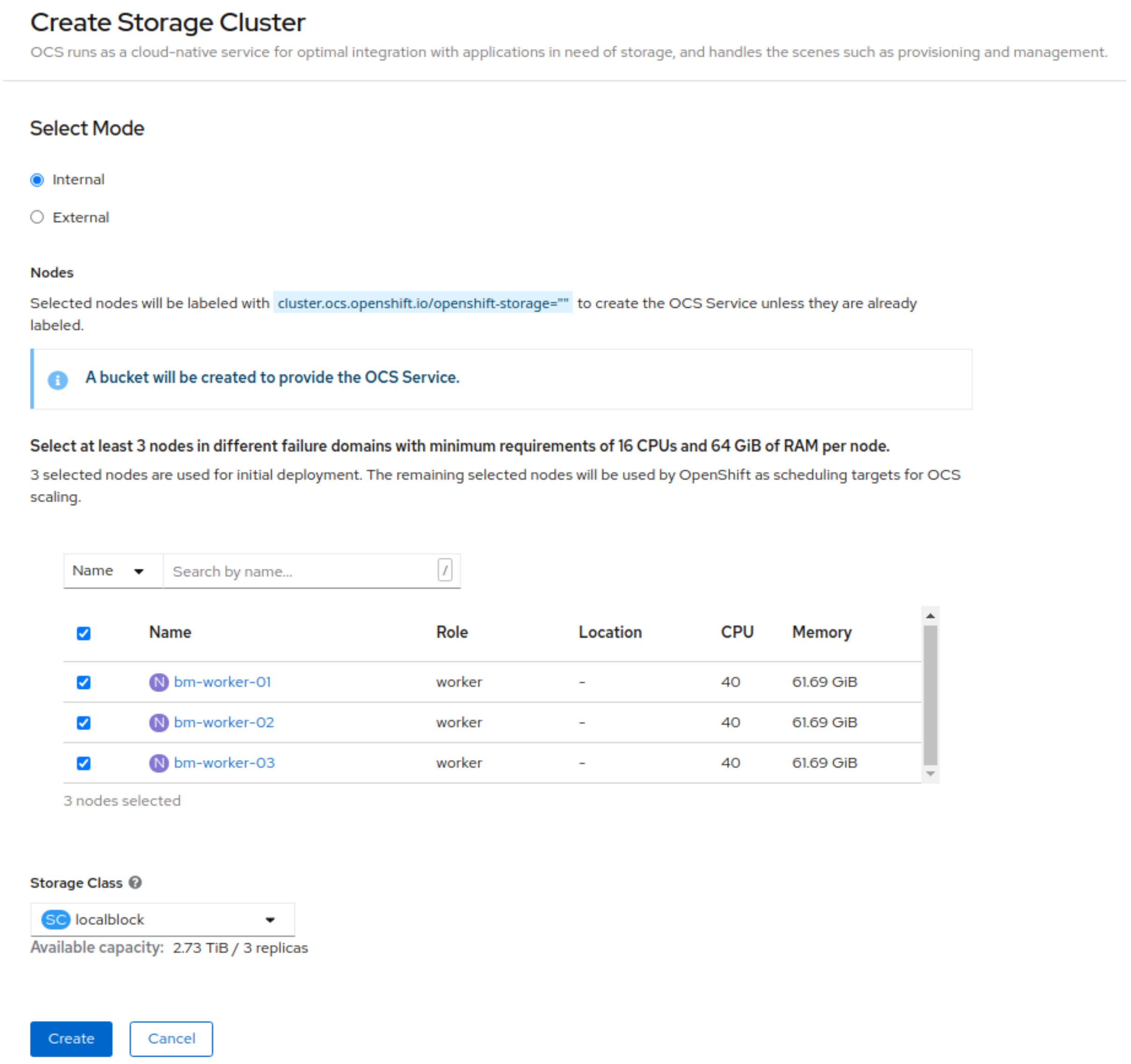
Details | YAML | Subscription | Events | All Instances | Storage Cluster | Backing Store | Bucket Class

OCS Cluster Services

Create OCS Cluster Service

No Operands Found

7. On the **Create Storage Cluster** page, ensure that the following options are selected:



Create Storage Cluster

OCS runs as a cloud-native service for optimal integration with applications in need of storage, and handles the scenes such as provisioning and management.

Select Mode

● Internal

○ External

Nodes

Selected nodes will be labeled with `cluster.ocs.openshift.io/openshift-storage=""` to create the OCS Service unless they are already labeled.

A bucket will be created to provide the OCS Service.

Select at least 3 nodes in different failure domains with minimum requirements of 16 CPUs and 64 GiB of RAM per node.

3 selected nodes are used for initial deployment. The remaining selected nodes will be used by OpenShift as scheduling targets for OCS scaling.

✓	Name	Role	Location	CPU	Memory
✓	N bm-worker-01	worker	-	40	61.69 GiB
✓	N bm-worker-02	worker	-	40	61.69 GiB
✓	N bm-worker-03	worker	-	40	61.69 GiB

3 nodes selected

Storage Class: SC localblock

Available capacity: 2.73 TiB / 3 replicas

- Leave **Select Mode** as **Internal**.
- In the **Nodes** section, for the use of OpenShift Container Storage service, select a minimum of three or a multiple of three worker nodes from the available list.
  It is recommended that the worker nodes are spread across three different physical nodes, racks or failure domains for high availability.

  NOTE

  - To find specific worker nodes in the cluster, you can filter nodes on the basis of Name or Label.
    - Name allows you to search by name of the node
    - Label allows you to search by selecting the predefined label
  - Ensure OpenShift Container Storage rack labels are aligned with physical racks in the datacenter to prevent a double node failure at the failure domain level.

  For minimum starting node requirements, see Resource requirements section in Planning guide.

- Select **localblock** from the **Storage Class** dropdown list.

8. Click **Create**.

NOTE

The **Create** button is enabled only after selecting a minimum of three worker nodes.

Upon successful deployment, a storage cluster with three storage devices gets created. These devices get distributed across three of the selected nodes. The configuration uses a replication factor of 3. To scale the initial cluster, see Scaling storage nodes .

Verification steps

See Verifying your OpenShift Container Storage installation .

CHAPTER 2. VERIFYING OPENSHIFT CONTAINER STORAGE DEPLOYMENT

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

2.1. VERIFYING THE STATE OF THE PODS

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

Procedure

1. Click **Workloads → Pods** from the left pane of the OpenShift Web Console.
2. Select **openshift-storage** from the **Project** drop down list.
   For more information on the expected number of pods for each component and how it varies depending on the number of nodes, see Table 2.1, "Pods corresponding to OpenShift Container storage cluster".

   NOTE

   When you need to override the cluster-wide default node selector for OpenShift Container Storage, you can perform the following steps through the command line interface:

   1. Specify a blank node selector for the **openshift-storage** namespace.

      ```
      $ oc annotate namespace openshift-storage openshift.io/node-selector=
      ```

   2. Delete the original pods generated by the **DaemonSets**.

      ```
      oc delete pod -l app=csi-cephfsplugin -n openshift-storage
      oc delete pod -l app=csi-rbdplugin -n openshift-storage
      ```

3. Verify that the following pods are in running and completed state by clicking on the **Running** and the **Completed** tabs:

Table 2.1. Pods corresponding to OpenShift Container storage cluster

Component	Corresponding pods
OpenShift Container Storage Operator	**ocs-operator-*** (1 pod on any worker node)
Rook-ceph Operator	**rook-ceph-operator-*** (1 pod on any worker node)

Component	Corresponding pods
Multicloud Object Gateway	• **noobaa-operator-*** (1 pod on any worker node) • **noobaa-core-*** (1 pod on any storage node) • **nooba-db-*** (1 pod on any storage node) • **noobaa-endpoint-*** (1 pod on any storage node)
MON	**rook-ceph-mon-*** (3 pods distributed across storage nodes)
MGR	**rook-ceph-mgr-*** (1 pod on any storage node)
MDS	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** (2 pods distributed across storage nodes)
RGW	**rook-ceph-rgw-ocs-storagecluster-cephobjectstore-*** (2 pods distributed across storage nodes)
CSI	• **cephfs** ◦ **csi-cephfsplugin-*** (1 pod on each worker node) ◦ **csi-cephfsplugin-provisioner-*** (2 pods distributed across storage nodes) • **rbd** ◦ **csi-rbdplugin-*** (1 pod on each worker node) ◦ **csi-rbdplugin-provisioner-*** (2 pods distributed across storage nodes)
rook-ceph-drain-canary	**rook-ceph-drain-canary-*** (1 pod on each storage node)
rook-ceph-crashcollector	**rook-ceph-crashcollector-*** (1 pod on each storage node)

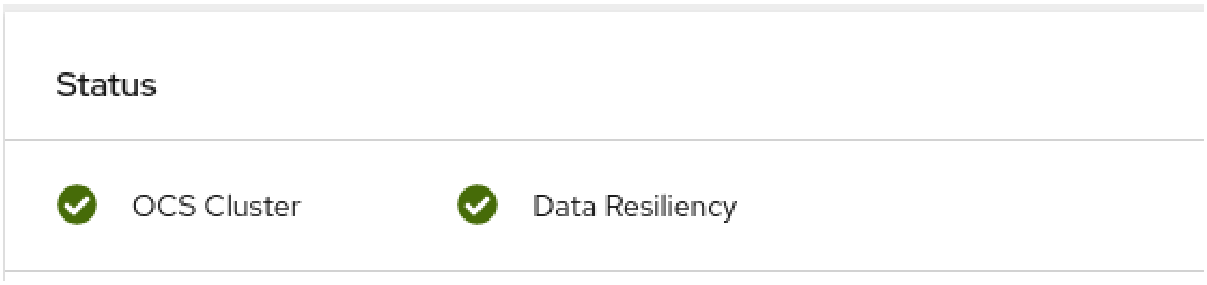
Component	Corresponding pods
OSD	• **rook-ceph-osd-*** (1 pod for each device) • **rook-ceph-osd-prepare-ocs-deviceset-*** (1 pod for each device)

2.2. VERIFYING THE OPENSHIFT CONTAINER STORAGE CLUSTER IS HEALTHY

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

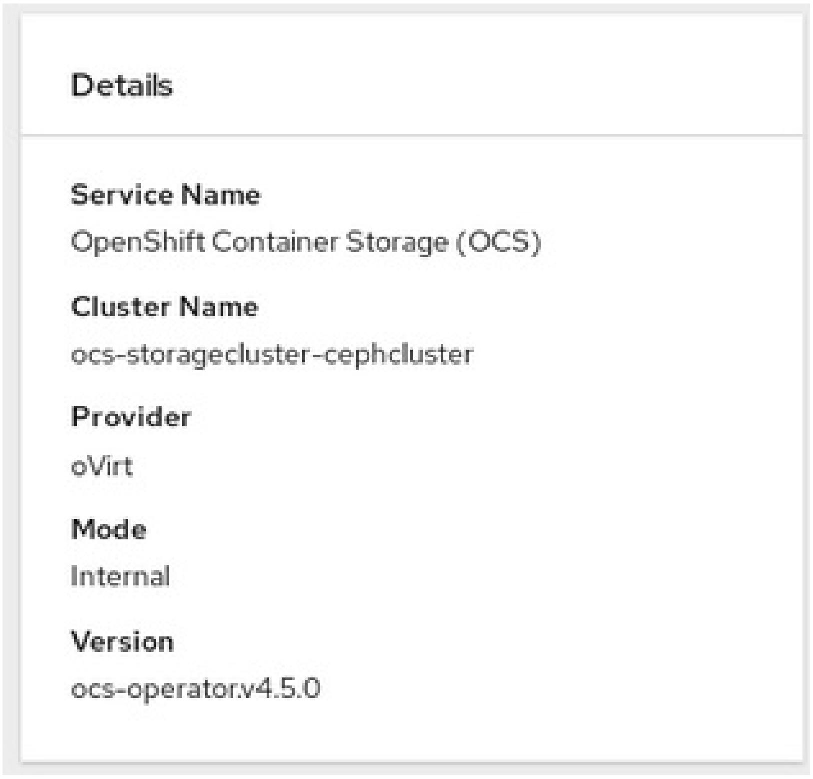
- Click **Home → Overview** from the left pane of the OpenShift Web Console and click **Persistent Storage** tab.
- In the **Status card**, verify that *OCS Cluster* has a green tick mark as shown in the following image:

  Figure 2.1. Health status card in Persistent Storage Overview Dashboard

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

Figure 2.2. Details card in Persistent Storage Overview Dashboard


Details

Service Name
OpenShift Container Storage (OCS)

Cluster Name
ocs-storagecluster-cephcluster

Provider
oVirt

Mode
Internal

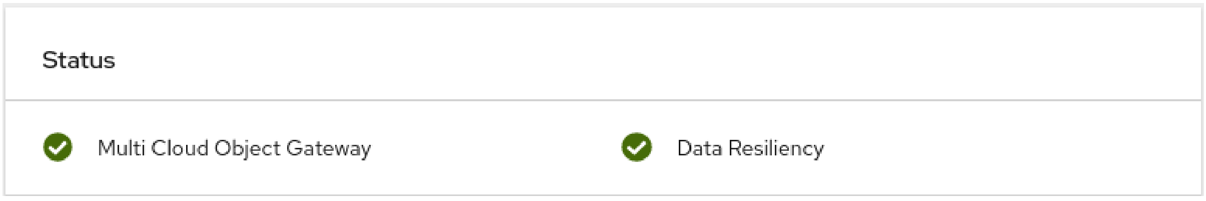
Version
ocs-operator.v4.5.0

2.3. VERIFYING THE MULTICLOUD OBJECT GATEWAY IS HEALTHY

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

- 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:

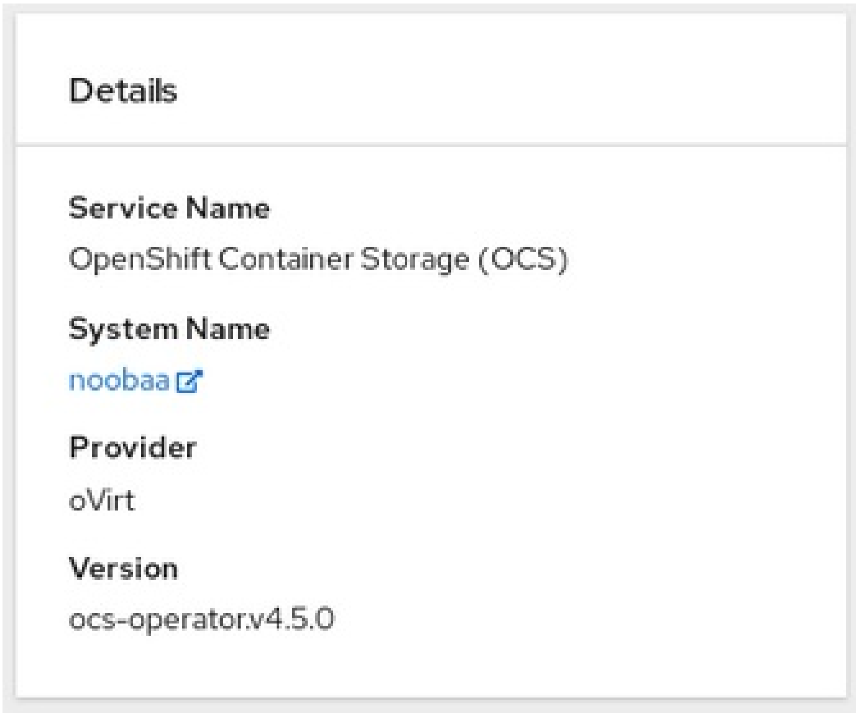
Figure 2.3. Health status card in Object Service Overview Dashboard


Status

✓ Multi Cloud Object Gateway ✓ Data Resiliency

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

Figure 2.4. Details card in Object Service Overview Dashboard



Details

Service Name
OpenShift Container Storage (OCS)

System Name
noobaa

Provider
oVirt

Version
ocs-operator.v4.5.0

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

To verify the storage classes exists in the cluster:

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

CHAPTER 3. UNINSTALLING OPENSHIFT CONTAINER STORAGE

3.1. UNINSTALLING OPENSHIFT CONTAINER STORAGE ON INTERNAL MODE

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 a 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 an unhealthy state, you should contact Red Hat Customer Support before uninstalling OpenShift Container Storage.
- Make sure that applications are not consuming persistent volume claims (PVCs) or object bucket claims (OBCs) using the storage classes provided by OpenShift Container Storage. PVCs and OBCs will be deleted during the uninstall process.

Procedure

1. Query for PVCs and OBCs that use the OpenShift Container Storage based storage class provisioners.
   For example :

   ```
   $ 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/ )' | grep -v noobaa-default-backing-store-noobaa-pvc
   ```

   ```
   $ 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
   ```

2. Follow these instructions to ensure that the PVCs and OBCs listed in the previous step are deleted.
   If you have created PVCs as a part of configuring the monitoring stack, cluster logging operator, or image registry, then you must perform the clean up steps provided in the following sections as required:

   - Section 3.2, “Removing monitoring stack from OpenShift Container Storage”
   - Section 3.3, “Removing OpenShift Container Platform registry from OpenShift Container Storage”
   - Section 3.4, “Removing the cluster logging operator from OpenShift Container Storage”
     For each of the remaining PVCs or OBCs, follow the steps mentioned below :

a. Determine the pod that is consuming the PVC or OBC.

b. Identify the controlling API object such as a **Deployment**, **StatefulSet**, **DaemonSet** , **Job**, or a custom controller.
   Each API 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 API object is not consuming PVC or OBC provided by OpenShift Container Storage. Either the object should be deleted or the storage should be replaced. Ask the owner of the project to make sure that it is safe to delete or modify the object.

   NOTE

   You can ignore the **noobaa** pods.

d. Delete the OBCs.

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

e. Delete any custom Bucket Class you have created.

```
$ oc get bucketclass -A  | grep -v noobaa-default-bucket-class
```

```
oc delete bucketclass <bucketclass name> -n <project-name>
```

f. If you have created any custom Multi Cloud Gateway backingstores, delete them.

   - List and note the backingstores.

```
for bs in $(oc get backingstore -o name -n openshift-storage | grep -v noobaa-default-backing-store); do echo "Found backingstore $bs"; echo "Its has the following pods running :"; echo "$(oc get pods -o name -n openshift-storage | grep $(echo ${bs} | cut -f2 -d/))"; done
```

   - Delete each of the backingstores listed above and confirm that the dependent resources also get deleted.

```
for bs in $(oc get backingstore -o name -n openshift-storage | grep -v noobaa-default-backing-store); do echo "Deleting Backingstore $bs"; oc delete -n openshift-storage $bs; done
```

   - If any of the backingstores listed above were based on the pv-pool, ensure that the corresponding pod and PVC are also deleted.

```
$ oc get pods -n openshift-storage | grep noobaa-pod | grep -v noobaa-default-backing-store-noobaa-pod
```

```
$ oc get pvc -n openshift-storage --no-headers | grep -v noobaa-db | grep noobaa-pvc | grep -v noobaa-default-backing-store-noobaa-pvc
```

   g. Delete the remaining PVCs listed in Step 1.

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

3. Delete the **StorageCluster** object and wait for the removal of the associated resources.

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

4. Delete the namespace and wait till the deletion is complete. You will need to switch to another project if openshift-storage is the active project.

   a. Switch to another namespace if openshift-storage is the active namespace.
      For example :

      ```
      $ oc project default
      ```

   b. Delete the openshift-storage namespace.

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

   c. Wait for approximately five minutes and confirm if the project is deleted successfully.

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

      Output:

      ```
      Error from server (NotFound): namespaces "openshift-storage" not found
      ```

      NOTE

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

5. 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.

   Confirm that the directory no longer exists

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

6. Delete the **openshift-storage.noobaa.io** storage class.

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

7. 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.

8. Confirm all PVs are deleted. If there is any PV left in the Released state, delete it.

```
# oc get pv | egrep 'ocs-storagecluster-ceph-rbd|ocs-storagecluster-cephfs'
```

```
# oc delete pv <pv name>
```

9. 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
cephclients.ceph.rook.io --wait=true --timeout=5m
```

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

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

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

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

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

3.3. REMOVING OPENSHIFT CONTAINER PLATFORM REGISTRY FROM OPENSHIFT CONTAINER STORAGE

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

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

Prerequisites

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

Procedure

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

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

Before editing

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

After editing

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

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

2. Delete the PVC.

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

3.4. REMOVING THE CLUSTER LOGGING OPERATOR FROM OPENSHIFT CONTAINER STORAGE

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

The PVCs that are created as a part of configuring cluster logging operator are in **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 the Curator schedule and the *Modifying retention time for Prometheus metrics data* sub section of Configuring persistent storage in the OpenShift Container Platform documentation for details.

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

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.

WARNING

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

Prerequisites

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

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

Procedure

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

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

      For example:

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

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

a. Click **Workloads → Pods**.

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

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

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

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

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.
- A storage class with provisioner **openshift-storage.rbd.csi.ceph.com** is available. In OpenShift Web Console, click **Storage → Storage Classes**to view available storage classes.

Procedure

1. In 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 **openshift-monitoring-config** Config Map using the following example.

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

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

Example **openshift-monitoring-config** Config Map

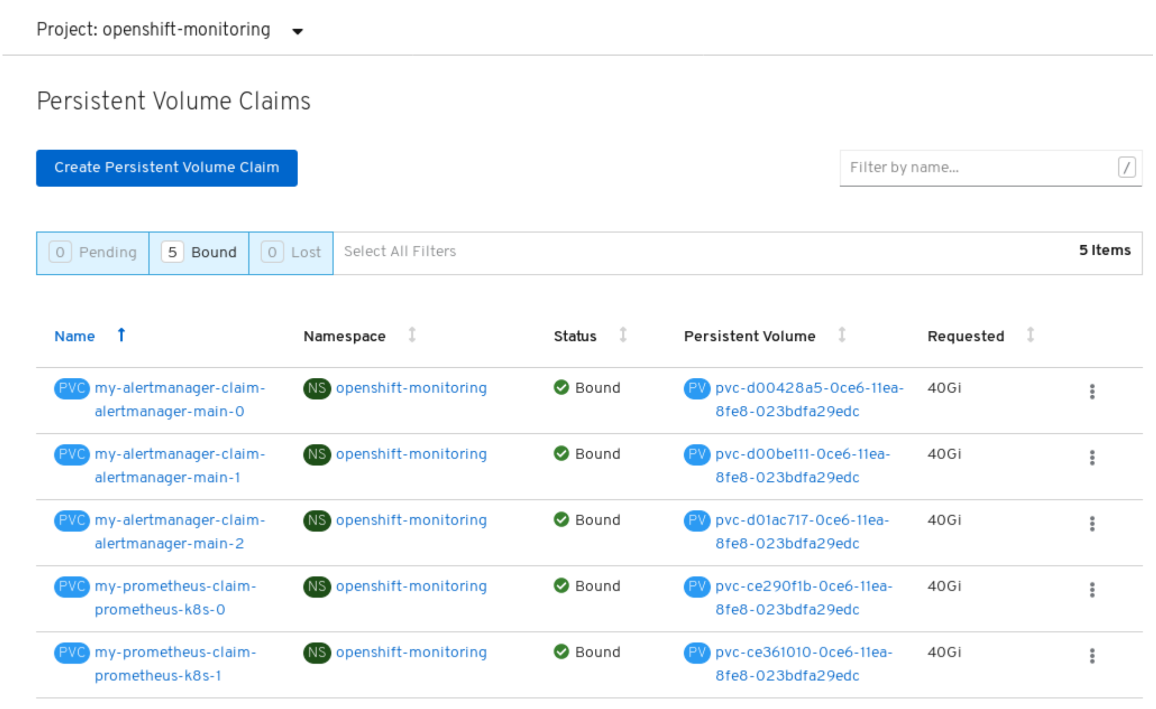
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
      prometheusK8s:
        retention: <time to retain monitoring files, e.g. 24h>
        volumeClaimTemplate:
          metadata:
            name: ocs-prometheus-claim
          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



Project: openshift-monitoring

Persistent Volume Claims

Create Persistent Volume Claim

Filter by name...

0 Pending | 5 Bound | 0 Lost | Select All Filters | 5 Items

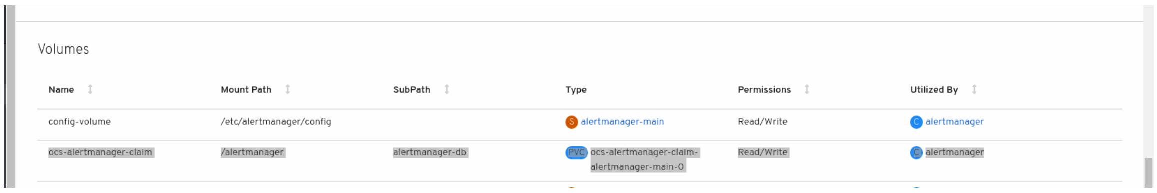
Name	Namespace	Status	Persistent Volume	Requested
PVC my-alertmanager-claim-alertmanager-main-0	NS openshift-monitoring	Bound	PV pvc-d00428a5-0ce6-11ea-8fe8-023bdfa29edc	40Gi
PVC my-alertmanager-claim-alertmanager-main-1	NS openshift-monitoring	Bound	PV pvc-d00be111-0ce6-11ea-8fe8-023bdfa29edc	40Gi
PVC my-alertmanager-claim-alertmanager-main-2	NS openshift-monitoring	Bound	PV pvc-d01ac717-0ce6-11ea-8fe8-023bdfa29edc	40Gi
PVC my-prometheus-claim-prometheus-k8s-0	NS openshift-monitoring	Bound	PV pvc-ce290f1b-0ce6-11ea-8fe8-023bdfa29edc	40Gi
PVC my-prometheus-claim-prometheus-k8s-1	NS openshift-monitoring	Bound	PV pvc-ce361010-0ce6-11ea-8fe8-023bdfa29edc	40Gi

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



Volumes

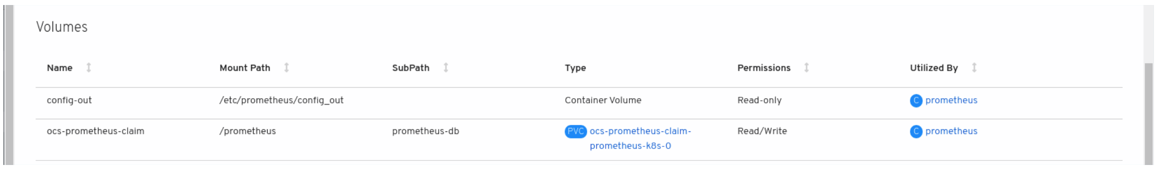
Name	Mount Path	SubPath	Type	Permissions	Utilized By
config-volume	/etc/alertmanager/config		S alertmanager-main	Read/Write	C alertmanager
ocs-alertmanager-claim	/alertmanager	alertmanager-db	PVC ocs-alertmanager-claim-alertmanager-main-0	Read/Write	C alertmanager

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



Volumes

Name	Mount Path	SubPath	Type	Permissions	Utilized By
config-out	/etc/prometheus/config_out		Container Volume	Read-only	C prometheus
ocs-prometheus-claim	/prometheus	prometheus-db	PVC ocs-prometheus-claim-prometheus-k8s-0	Read/Write	C prometheus

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 Cluster logging curator in the OpenShift Container Platform documentation for details.

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

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

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: ocs-storagecluster-ceph-rbd
        size: 200G
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      replicas: 1
```

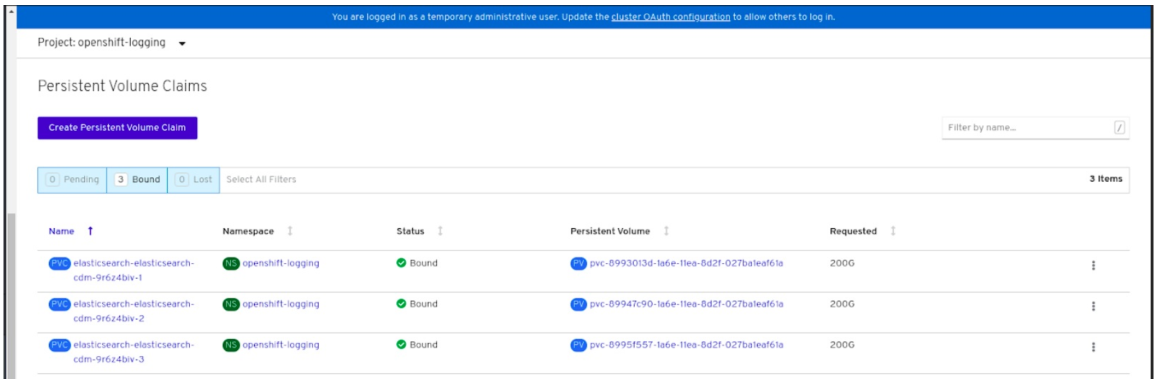
```
curation:
  type: "curator"
  curator:
    schedule: "30 3 * * *"
collection:
  logs:
    type: "fluentd"
    fluentd: {}
```

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



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: openshift-logging

Persistent Volume Claims

Create Persistent Volume Claim

Filter by name...

0 Pending | 3 Bound | 0 Lost | Select All Filters — 3 Items

Name	Namespace	Status	Persistent Volume	Requested
PVC elasticsearch-elasticsearch-cdm-9r6z4biv-1	NS openshift-logging	Bound	PV pvc-8993013d-1a6e-11ea-8d2f-027ba1eaf61a	200G
PVC elasticsearch-elasticsearch-cdm-9r6z4biv-2	NS openshift-logging	Bound	PV pvc-89947c90-1a6e-11ea-8d2f-027ba1eaf61a	200G
PVC elasticsearch-elasticsearch-cdm-9r6z4biv-3	NS openshift-logging	Bound	PV pvc-8995f557-1a6e-11ea-8d2f-027ba1eaf61a	200G

2. Verify that the new cluster logging is being used.
   a. Click **Workload → Pods**.
   b. Set the Project to **openshift-logging**.
   c. Verify that the new **elasticsearch-*** pods appear with a state of **Running**.
   d. Click the new **elasticsearch-*** pod to view pod details.
   e. Scroll down to **Volumes** and verify that the elasticsearch volume has a **Type** that matches your new Persistent Volume Claim, for example, **elasticsearch-elasticsearch-cdm-9r624biv-3**.
   f. Click the Persistent Volume Claim name and verify the storage class name in the PersistenVolumeClaim Overview page.

NOTE

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

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

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

For more details, see Curation of Elasticsearch Data .

NOTE

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

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

        NOTE

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

    d. Select the size of the desired storage capacity.

        NOTE

        You 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

6.1. REQUIREMENTS FOR SCALING STORAGE NODES

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

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

WARNING

Always ensure that you have plenty of storage capacity.

If storage ever fills completely, it is not possible to add capacity or delete or migrate content away from the storage to free up space. Completely full storage is very difficult to recover.

Capacity alerts are issued when cluster storage capacity reaches 75% (near-full) and 85% (full) of total capacity. Always address capacity warnings promptly, and review your storage regularly to ensure that you do not run out of storage space.

If you do run out of storage space completely, contact Red Hat Customer Support.

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

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

Prerequisites

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

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

Procedure

1. To add storage capacity to OpenShift Container Platform nodes with OpenShift Container Storage installed, you need to
   a. Find the unique **by-id** identifier for available devices that you want to add, that is, a minimum of one device per worker node. You can follow the procedure for finding available storage devices in the respective deployment guide.

      NOTE

      Make sure you perform this process for all the existing nodes (minimum of 3) for which you want to add storage.

   b. Add the unique device ID to the **LocalVolume** custom resource (CR).

```
$ oc edit -n local-storage localvolume local-block
```

Example output:

```
spec:
  logLevel: Normal
  managementState: Managed
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: cluster.ocs.openshift.io/openshift-storage
        operator: In
        values:
        - ""
  storageClassDevices:
  - devicePaths:
    - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402P51P0GGN
    - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402LM1P0GGN
    - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402M21P0GGN
    - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402B71P0GGN    # newly
added device by-id
    - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402A31P0GGN    # newly
added device by-id
    - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402Q71P0GGN    # newly
added device by-id
    storageClassName: localblock
    volumeMode: Block
```

Make sure to save the changes after editing the CR.

Example output:

```
localvolume.local.storage.openshift.io/local-block edited
```

You can see in this CR that new devices using **by-id** have been added. Each new device maps to one NVMe device on the three worker nodes.

- **nvme-INTEL_SSDPE2KX010T7_PHLF733402B71P0GGN**
- **nvme-INTEL_SSDPE2KX010T7_PHLF733402A31P0GGN**
- **nvme-INTEL_SSDPE2KX010T7_PHLF733402Q71P0GGN**

2. Display the newly created PVs with **storageclass** name used in **localVolume** CR.

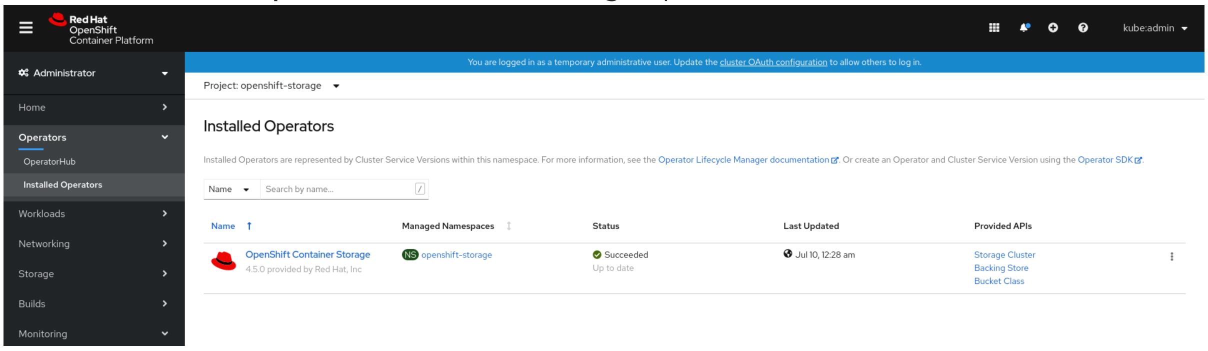
```
$ oc get pv | grep localblock | grep Available
```

Example output:

```
local-pv-5ee61dcc   931Gi   RWO     Delete  Available   localblock     2m35s
local-pv-b1fa607a   931Gi   RWO     Delete  Available   localblock     2m27s
local-pv-e971c51d   931Gi   RWO     Delete  Available   localblock     2m22s
...
```

There are three more available PVs of same size which will be used for new OSDs.

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



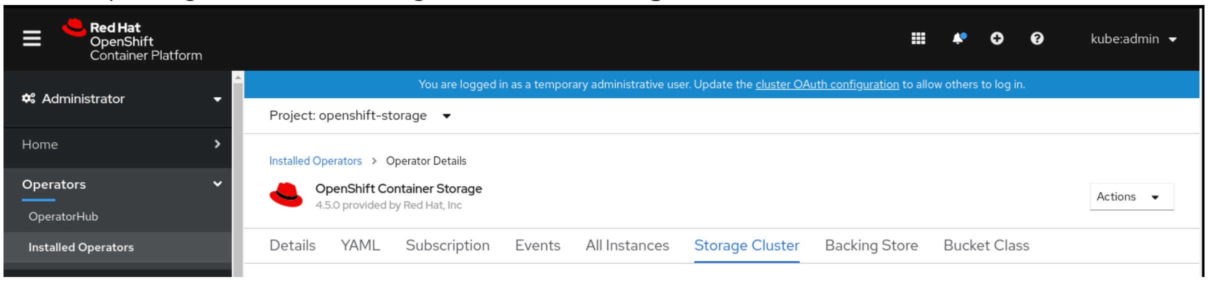
You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Installed Operators

Installed Operators are represented by Cluster Service Versions within this namespace. For more information, see the Operator Lifecycle Manager documentation. Or create an Operator and Cluster Service Version using the Operator SDK.

Name	Managed Namespaces	Status	Last Updated	Provided APIs
OpenShift Container Storage 4.5.0 provided by Red Hat, Inc	NS openshift-storage	Succeeded Up to date	Jul 10, 12:28 am	Storage Cluster Backing Store Bucket Class

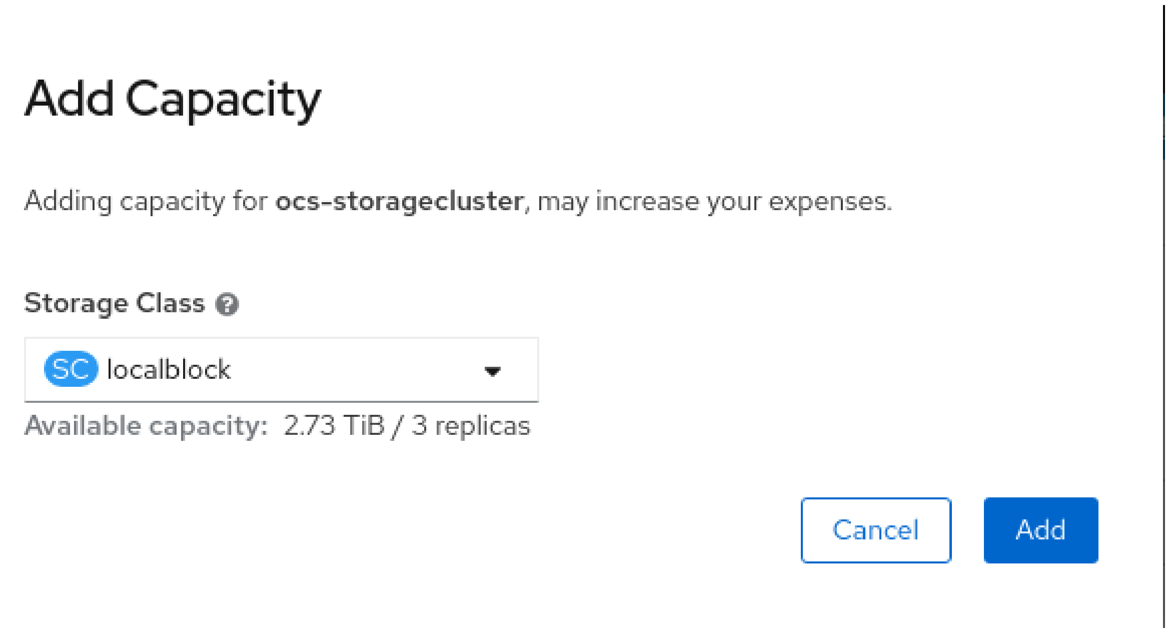
7. In the top navigation bar, scroll right and click **Storage Cluster** tab.



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

OpenShift Container Storage
4.5.0 provided by Red Hat, Inc

8. The visible list should have only one item. Click (⋮) on the far right to extend the options menu.
9. Select **Add Capacity** from the options menu.


Add Capacity

Adding capacity for **ocs-storagecluster**, may increase your expenses.

Storage Class

SC localblock

Available capacity: 2.73 TiB / 3 replicas

From this dialog box, set the **Storage Class** name to the name used in the **localVolume** CR. Available Capacity displayed is based on the local disks available in storage class.

10. Once you are done with your setting, click **Add**. You might need to wait a couple of minutes for the storage cluster to reach **Ready** state.

11. Verify that the new OSDs and their corresponding new PVCs are created.

```
$ oc get -n openshift-storage pods -l app=rook-ceph-osd
```

Example output:

```
NAME                              READY   STATUS    RESTARTS   AGE
rook-ceph-osd-0-77c4fdb758-qshw4   1/1     Running   0          1h
rook-ceph-osd-1-8645c5fbb6-656ks   1/1     Running   0          1h
rook-ceph-osd-2-86895b854f-r4gt6   1/1     Running   0          1h
rook-ceph-osd-3-dc7f787dd-gdnsz    1/1     Running   0          10m
rook-ceph-osd-4-554b5c46dd-hbf9t   1/1     Running   0          10m
rook-ceph-osd-5-5cf94c4448-k94j6   1/1     Running   0          10m
```

In the above example, osd-3, osd-4, and osd-5 are the newly added pods to the OpenShift Container Storage cluster.

```
$ oc get pvc -n openshift-storage |grep localblock
```

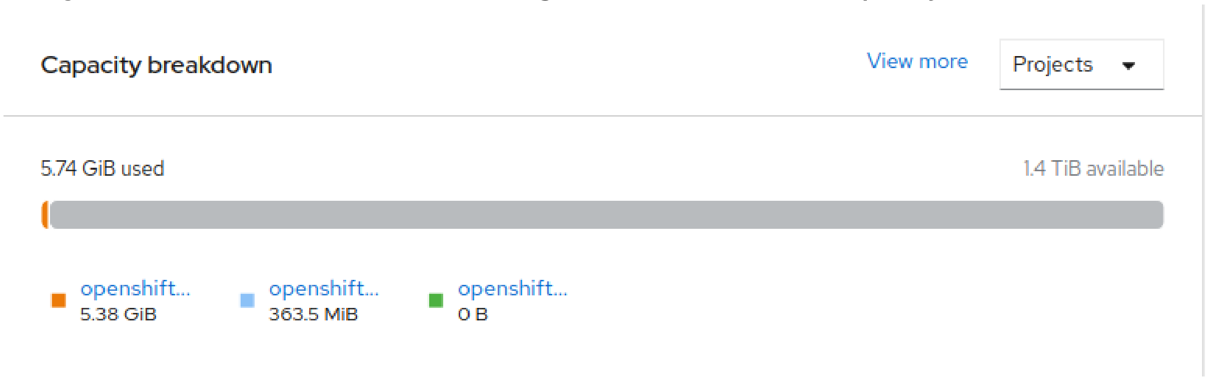
Example output:

```
ocs-deviceset-0-0-qc29m   Bound   local-pv-fc5562d3   931Gi   RWO  localblock  1h
ocs-deviceset-0-1-qdmrl   Bound   local-pv-b1fa607a   931Gi   RWO  localblock  10m
ocs-deviceset-1-0-mpwmk   Bound   local-pv-58cdd0bc   931Gi   RWO  localblock  1h
ocs-deviceset-1-1-85892   Bound   local-pv-e971c51d   931Gi   RWO  localblock  10m
ocs-deviceset-2-0-rll47   Bound   local-pv-29d8ad8d   931Gi   RWO  localblock  1h
ocs-deviceset-2-1-cgth2   Bound   local-pv-5ee61dcc   931Gi   RWO  localblock  10m
```

In the above example, we see three new PVCs are created.

Verification steps

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



Capacity breakdown

5.74 GiB used

1.4 TiB available

🟧 openshift...	5.38 GiB
🟦 openshift...	363.5 MiB
🟩 openshift...	0 B

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.

6.3. SCALING OUT STORAGE CAPACITY BY ADDING NEW NODES

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

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

6.3.1. Adding a node using a local storage device

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

Prerequisites

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

Procedure

1. Create a new VM on Red Hat Virtualization with the required infrastructure. See Platform requirements.
2. Create a new OpenShift Container Platform worker node using the new VM.
3. Check for certificate signing requests (CSRs) related to OpenShift Container Storage that are in **Pending** state:

```
$ oc get csr
```

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

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

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

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

   From User interface

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

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

   From Command line interface

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

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

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 6.3.2, "Verifying the addition of a new node" .

6.3.2. Verifying the addition of a new node

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

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

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

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

6.3.3. Scaling up storage capacity

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

CHAPTER 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 at Download RedHat OpenShift Container Storage page .

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 Download RedHat OpenShift Container Storage page .

Procedure

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

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

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

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

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

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

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

You can also add storage resources using a YAML:

1. Create a secret with the credentials:

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

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

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

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

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: noobaa
spec:
  awsS3:
    secret:
      name: <backingstore-secret-name>
      namespace: noobaa
    targetBucket: <bucket-name>
  type: <backing-store-type>
```

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

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

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

7.3.2. Creating an S3 compatible NooBaa backingstore

Procedure

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

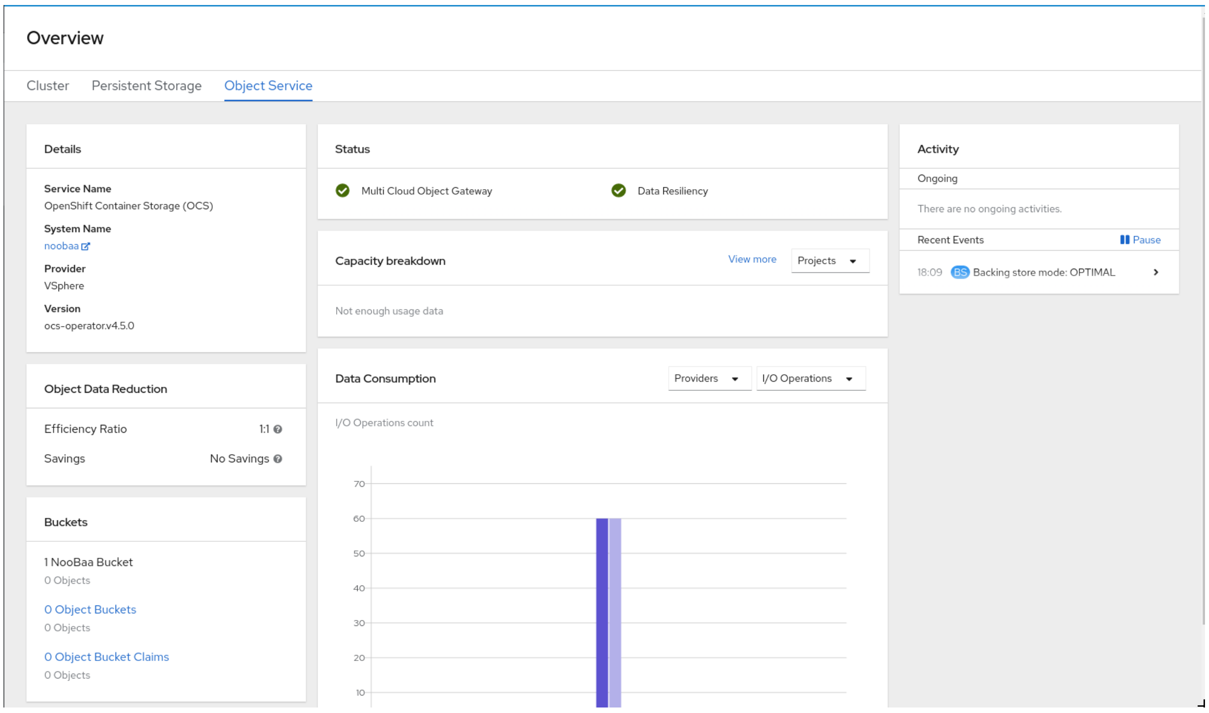
a. Replace **<backingstore-secret-name>** with the name of the secret that was created with **CephObjectStore** in the previous step.

b. Replace **<bucket-name>** with an existing RGW bucket name. This argument tells NooBaa 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:



Overview

Cluster | Persistent Storage | Object Service

Details

Service Name
OpenShift Container Storage (OCS)

System Name
noobaa

Provider
VSphere

Version
ocs-operator.v4.5.0

Object Data Reduction

Efficiency Ratio 1:1

Savings No Savings

Buckets

1 NooBaa Bucket
0 Objects

0 Object Buckets
0 Objects

0 Object Bucket Claims
0 Objects

Status

✓ Multi Cloud Object Gateway ✓ Data Resiliency

Capacity breakdown

Not enough usage data

Data Consumption

I/O Operations count

Activity

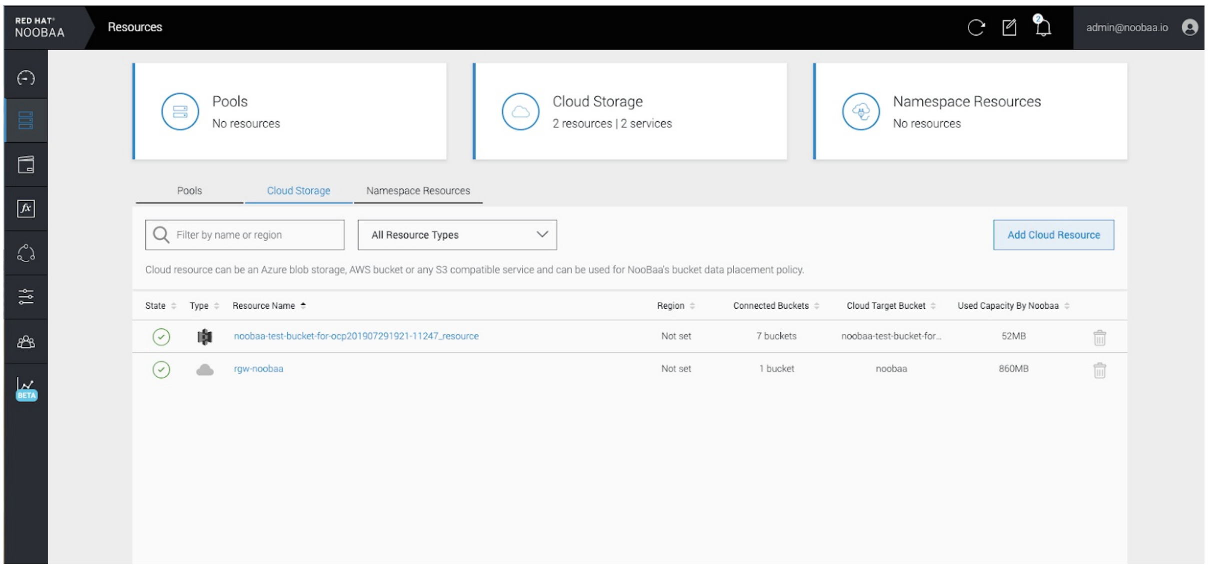
Ongoing

There are no ongoing activities.

Recent Events

18:09 BS Backing store mode: OPTIMAL

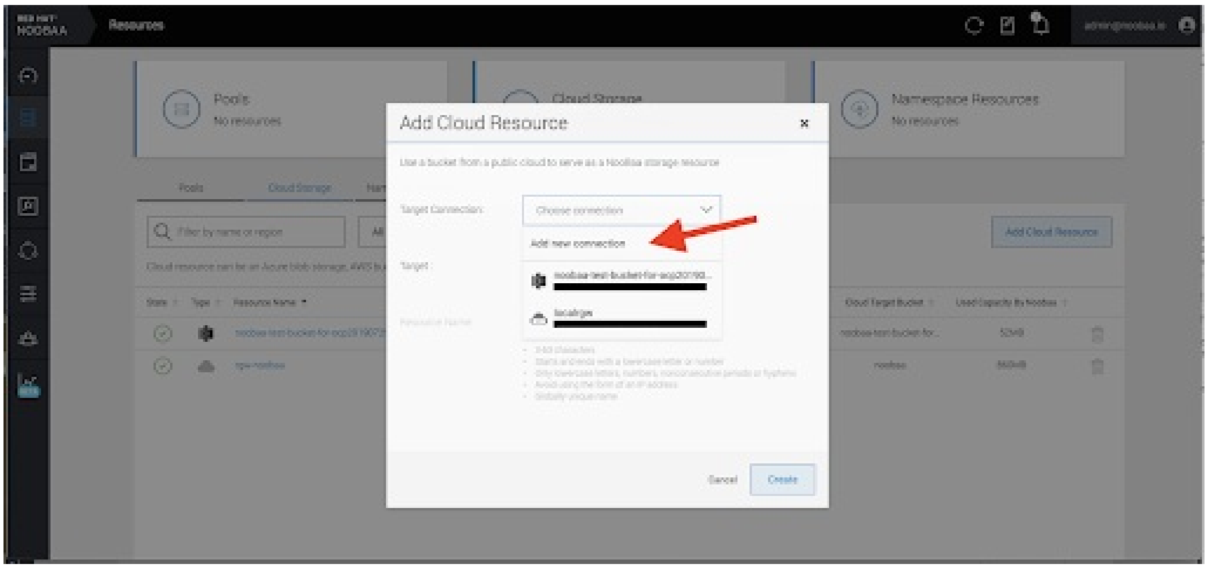
2. Select the **Resources** tab in the left, highlighted below. From the list that populates, select **Add Cloud Resource**:



Cloud resource can be an Azure blob storage, AWS bucket or any S3 compatible service and can be used for NooBaa's bucket data placement policy.

State	Type	Resource Name	Region	Connected Buckets	Cloud Target Bucket	Used Capacity By Noobaa
✓		noobaa-test-bucket-for-ocp201907291921-11247_resource	Not set	7 buckets	noobaa-test-bucket-for...	52MB
✓		rgw-noobaa	Not set	1 bucket	noobaa	860MB

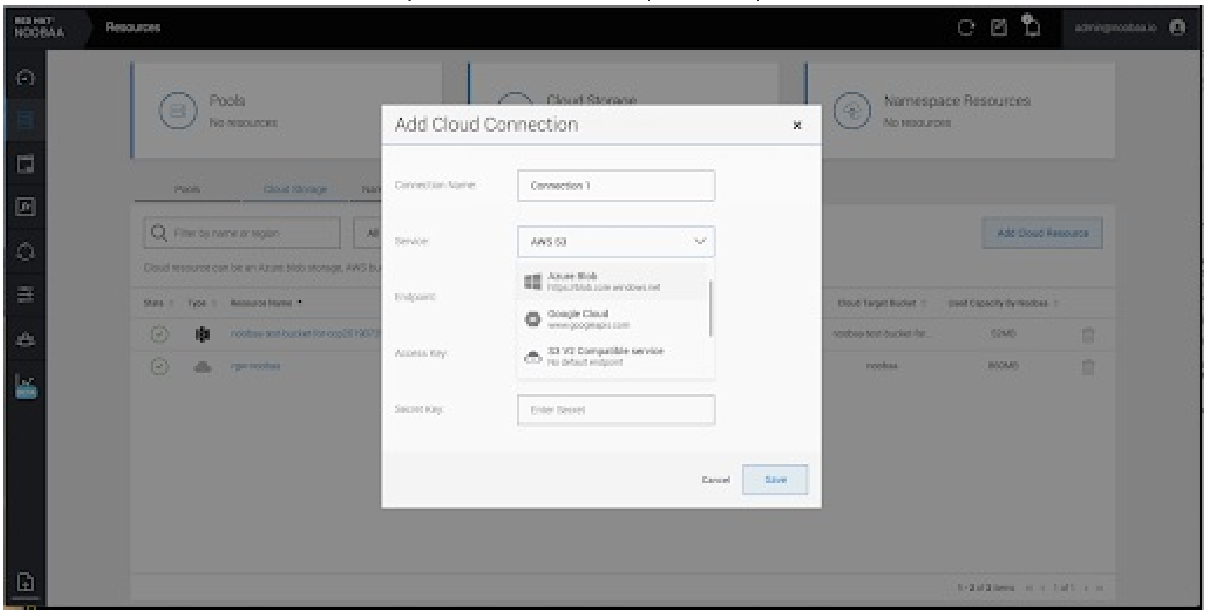
3. Select **Add new connection**:



Use a bucket from a public cloud to serve as a NooBaa storage resource

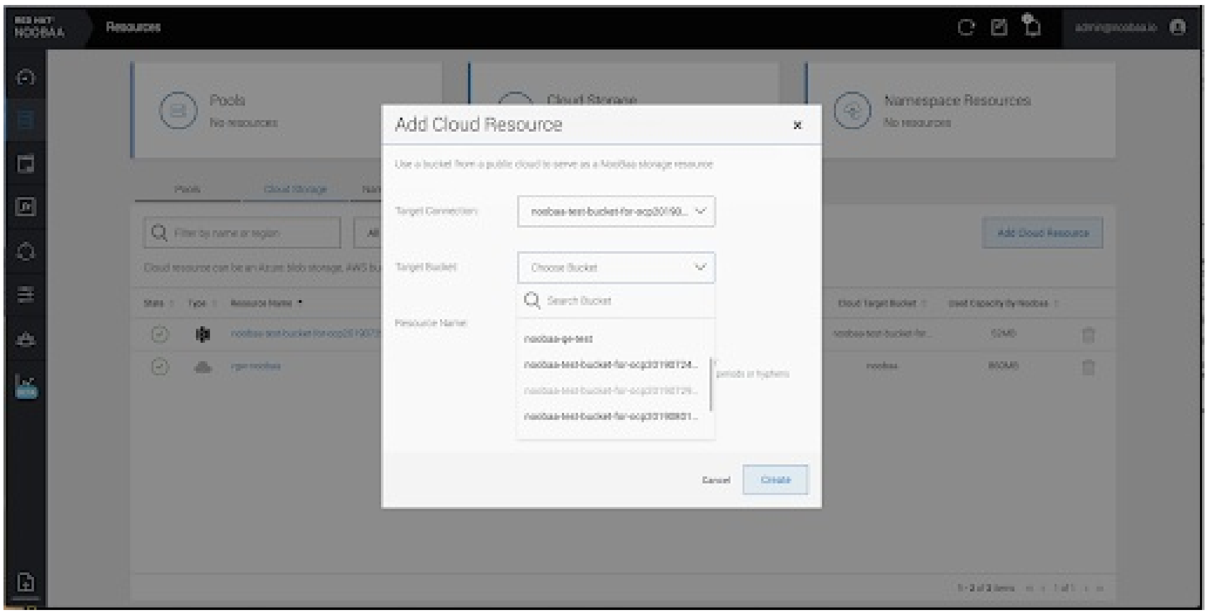
noobaa-test-bucket-for-ocp20190...
[redacted]
localrgw
[redacted]

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



Connection Name: Connection 1
Service: AWS S3

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.

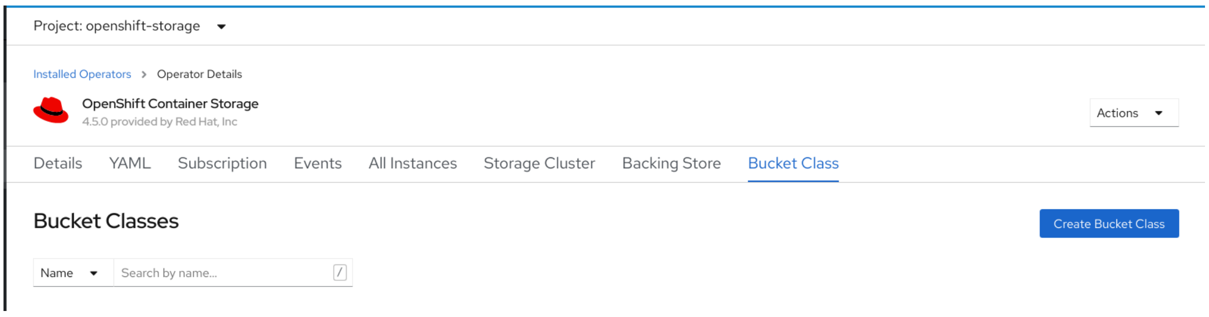
Procedure

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

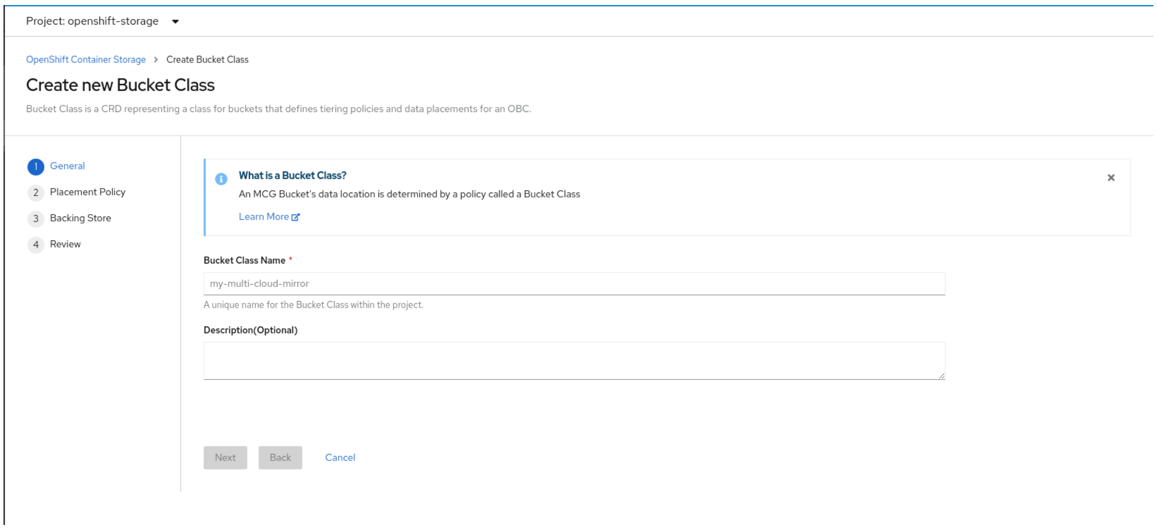


4. Click **Create Bucket Class**.

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



Project: openshift-storage

OpenShift Container Storage > Create Bucket Class

Create new Bucket Class

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

1 General
2 Placement Policy
3 Backing Store
4 Review

What is a Bucket Class?
An MCG Bucket's data location is determined by a policy called a Bucket Class
Learn More

Bucket Class Name *
my-multi-cloud-mirror
A unique name for the Bucket Class within the project.

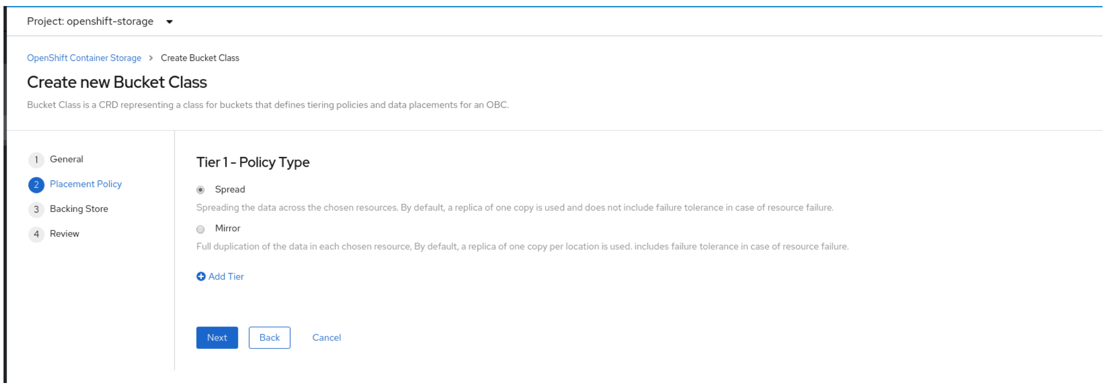
Description(Optional)

Next Back Cancel

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



Project: openshift-storage

OpenShift Container Storage > Create Bucket Class

Create new Bucket Class

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

1 General
2 Placement Policy
3 Backing Store
4 Review

Tier 1 - Policy Type

● Spread
Spreading the data across the chosen resources. By default, a replica of one copy is used and does not include failure tolerance in case of resource failure.

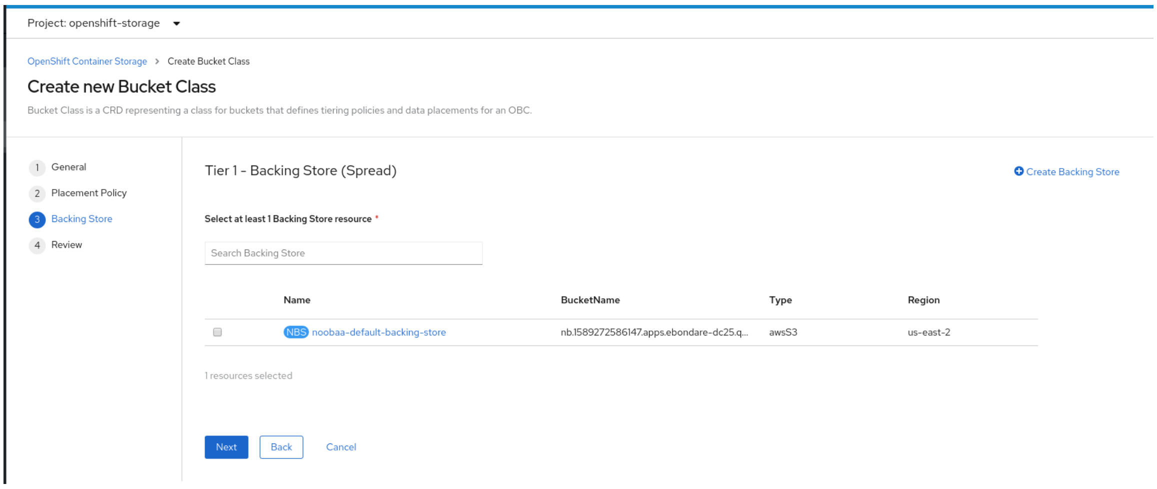
○ Mirror
Full duplication of the data in each chosen resource, By default, a replica of one copy per location is used. includes failure tolerance in case of resource failure.

Add Tier

Next Back Cancel

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



Project: openshift-storage

OpenShift Container Storage > Create Bucket Class

Create new Bucket Class

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

1 General
2 Placement Policy
3 Backing Store
4 Review

Tier 1 - Backing Store (Spread)

Create Backing Store

Select at least 1 Backing Store resource *

Search Backing Store

	Name	BucketName	Type	Region
☐	NBS noobaa-default-backing-store	nb.1589272586147.apps.ebondare-dc25.q...	awsS3	us-east-2

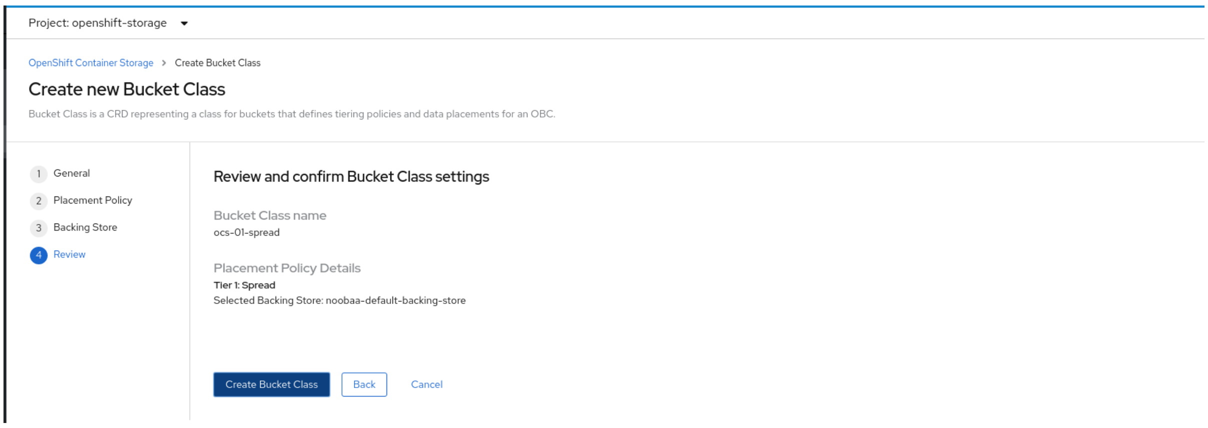
1 resources selected

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



Project: openshift-storage

OpenShift Container Storage > Create Bucket Class

Create new Bucket Class

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

1 General
2 Placement Policy
3 Backing Store
4 Review

Review and confirm Bucket Class settings

Bucket Class name
ocs-01-spread

Placement Policy Details
Tier 1: Spread
Selected Backing Store: noobaa-default-backing-store

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.

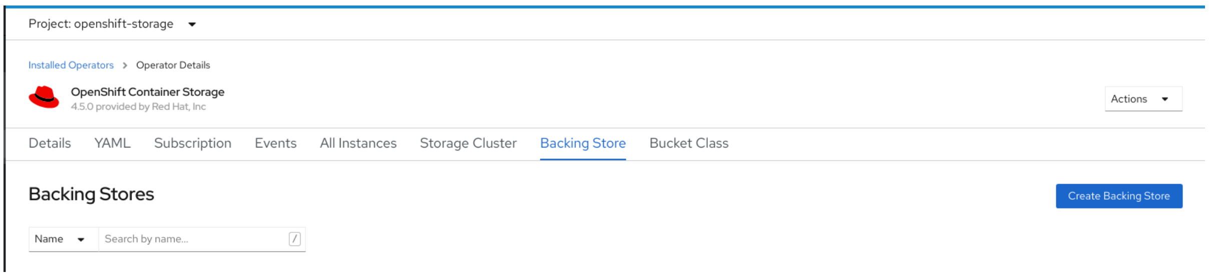
Procedure

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.

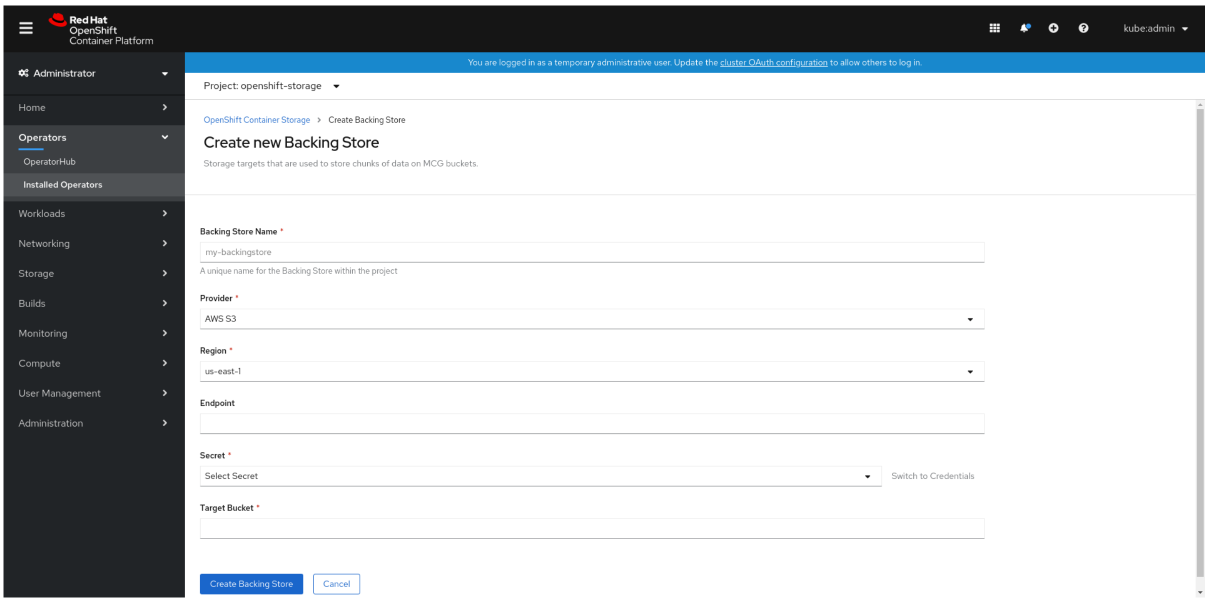
3. On the OpenShift Container Storage Operator page, scroll right and click the **Backing Store** tab.

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



4. Click **Create Backing Store**.

Figure 7.7. Create Backing Store page



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Create new Backing Store

Storage targets that are used to store chunks of data on MCG buckets.

Backing Store Name *: my-backingstore

A unique name for the Backing Store within the project

Provider *: AWS S3

Region *: us-east-1

Endpoint

Secret *: Select Secret

Target Bucket *

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.

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
```

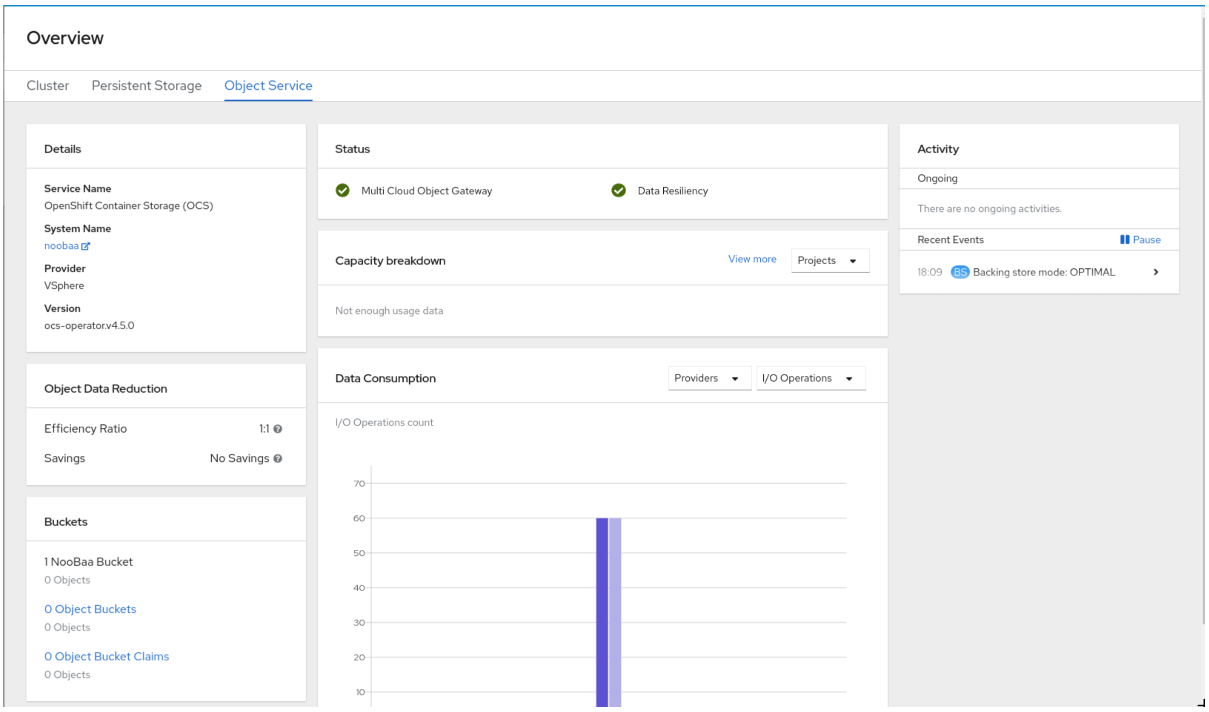
2. Add the following lines to your standard Object Bucket Claim (OBC):

```
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:



Overview

Cluster | Persistent Storage | Object Service

Details

Service Name
OpenShift Container Storage (OCS)
System Name
noobaa
Provider
VSphere
Version
ocs-operator.v4.5.0

Object Data Reduction

Efficiency Ratio 1:1
Savings No Savings

Buckets

1 NooBaa Bucket
0 Objects
0 Object Buckets
0 Objects
0 Object Bucket Claims
0 Objects

Status

Multi Cloud Object Gateway
Data Resiliency

Capacity breakdown | View more | Projects

Not enough usage data

Data Consumption | Providers | I/O Operations

I/O Operations count

Activity

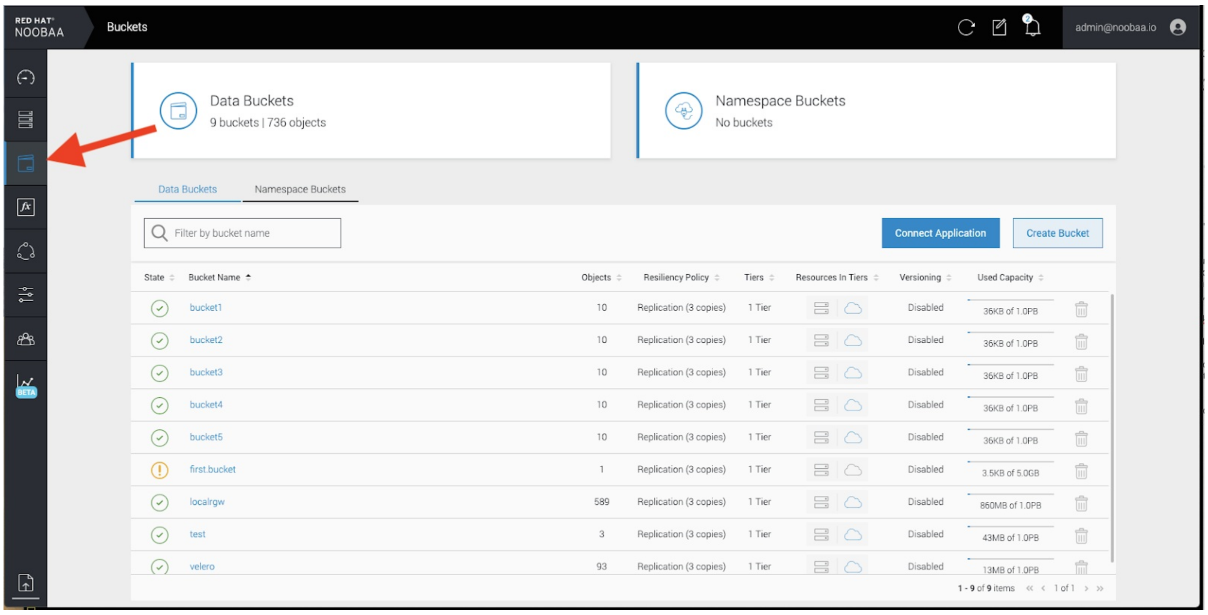
Ongoing

There are no ongoing activities.

Recent Events | Pause

18:09 BS Backing store mode: OPTIMAL

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

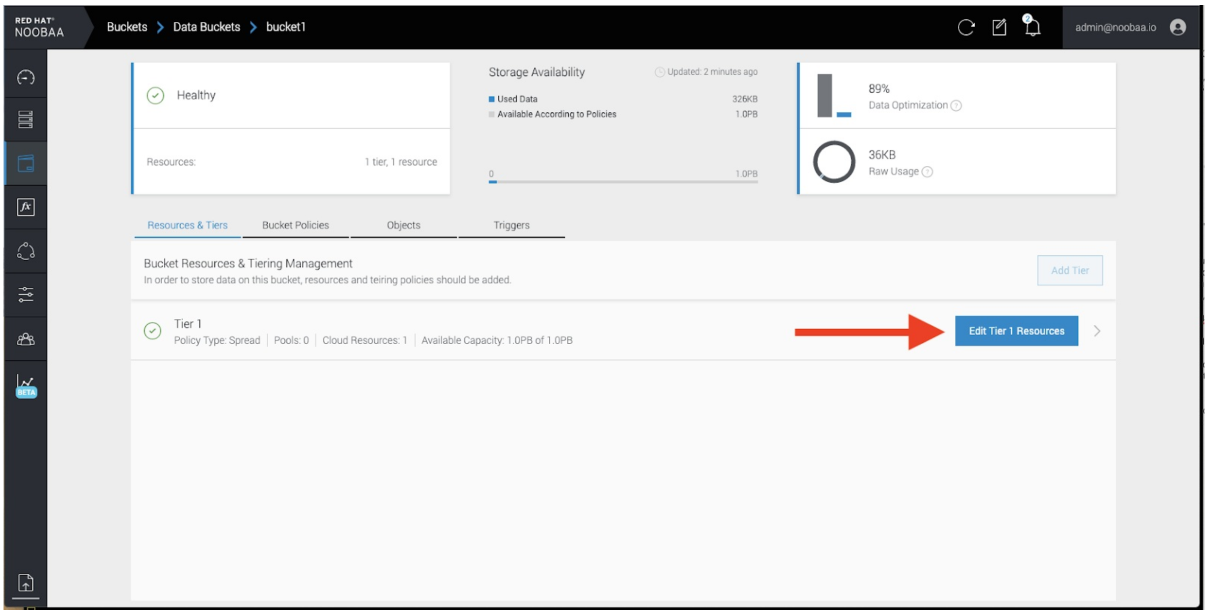


State	Bucket Name	Objects	Resiliency Policy	Tiers	Resources In Tiers	Versioning	Used Capacity
✓	bucket1	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket2	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket3	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket4	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket5	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
!	first.bucket	1	Replication (3 copies)	1 Tier		Disabled	3.5KB of 5.0GB
✓	localrgw	589	Replication (3 copies)	1 Tier		Disabled	860MB of 1.0PB
✓	test	3	Replication (3 copies)	1 Tier		Disabled	43MB of 1.0PB
✓	velero	93	Replication (3 copies)	1 Tier		Disabled	13MB of 1.0PB

1 - 9 of 9 items 1 of 1

3. Cick the bucket you want to update.

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



Healthy

Resources: 1 tier, 1 resource

Storage Availability — Updated: 2 minutes ago

Used Data 326KB

Available According to Policies 1.0PB

0 1.0PB

89% Data Optimization

36KB Raw Usage

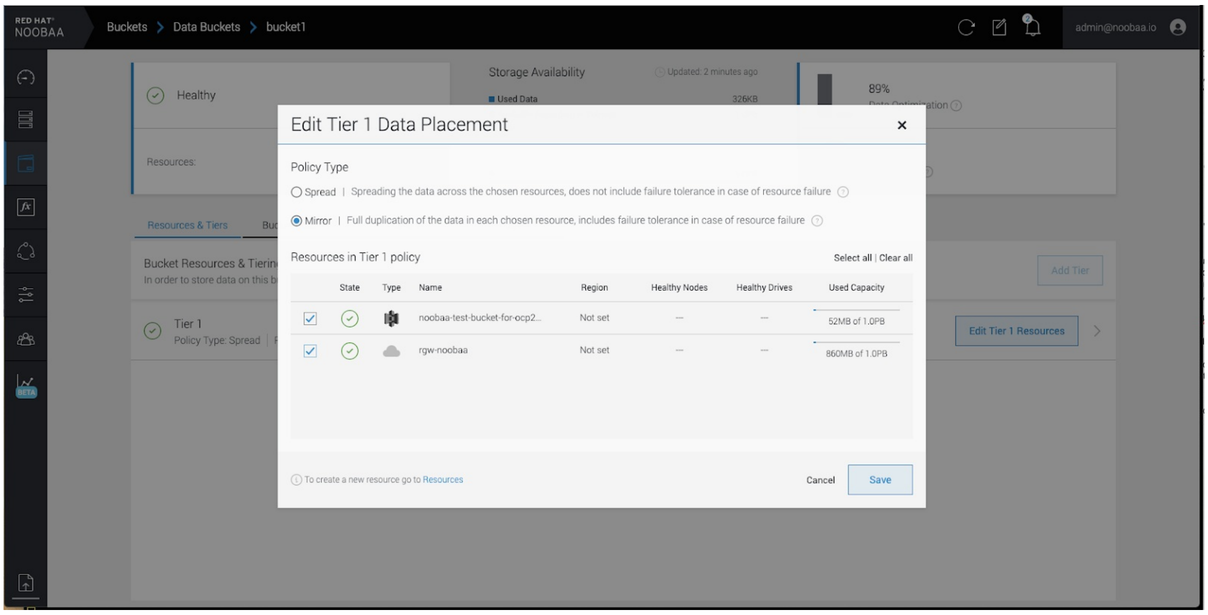
Bucket Resources & Tiering Management

In order to store data on this bucket, resources and teiring policies should be added.

Tier 1

Policy Type: Spread | Pools: 0 | Cloud Resources: 1 | Available Capacity: 1.0PB of 1.0PB

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:



Edit Tier 1 Data Placement

Policy Type

Spread | Spreading the data across the chosen resources, does not include failure tolerance in case of resource failure

[✓ Mirror] | Full duplication of the data in each chosen resource, includes failure tolerance in case of resource failure

Resources in Tier 1 policy — Select all | Clear all

	State	Type	Name	Region	Healthy Nodes	Healthy Drives	Used Capacity
✓	✓		noobaa-test-bucket-for-ocp2...	Not set	---	---	52MB of 1.0PB
✓	✓		rgw-noobaa	Not set	---	---	860MB of 1.0PB

To create a new resource go to Resources

Cancel | Save

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.

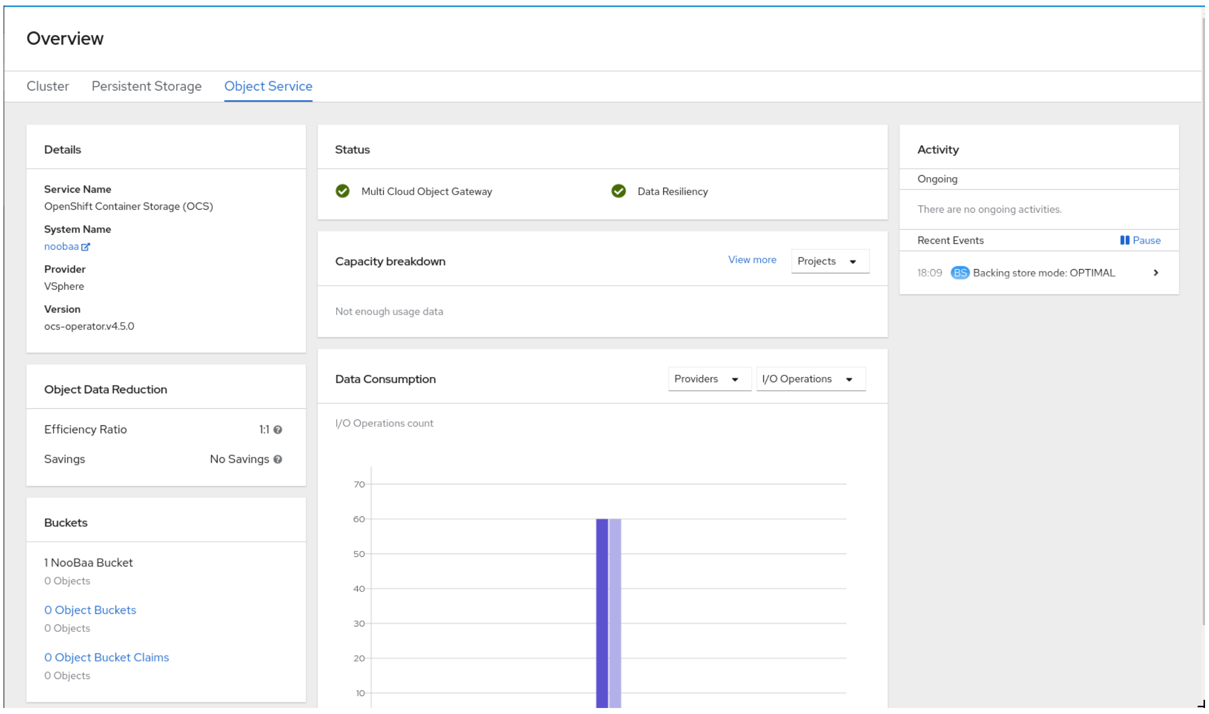
7.5.3. Creating an AWS S3 user in the Multicloud Object Gateway

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:



Details

Service Name
OpenShift Container Storage (OCS)
System Name
noobaa
Provider
VSphere
Version
ocs-operator.v4.5.0

Status

Multi Cloud Object Gateway
Data Resiliency

Activity

Ongoing
There are no ongoing activities.
Recent Events
Pause
18:09 BS Backing store mode: OPTIMAL

Capacity breakdown
View more
Projects
Not enough usage data

Object Data Reduction

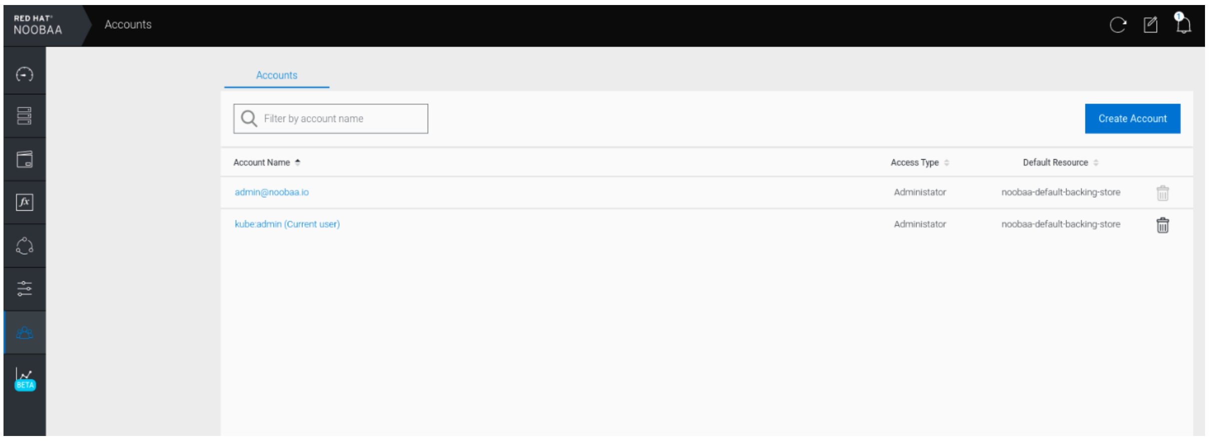
Efficiency Ratio	1:1
Savings	No Savings

Buckets

1 NooBaa Bucket
0 Objects
0 Object Buckets
0 Objects
0 Object Bucket Claims
0 Objects

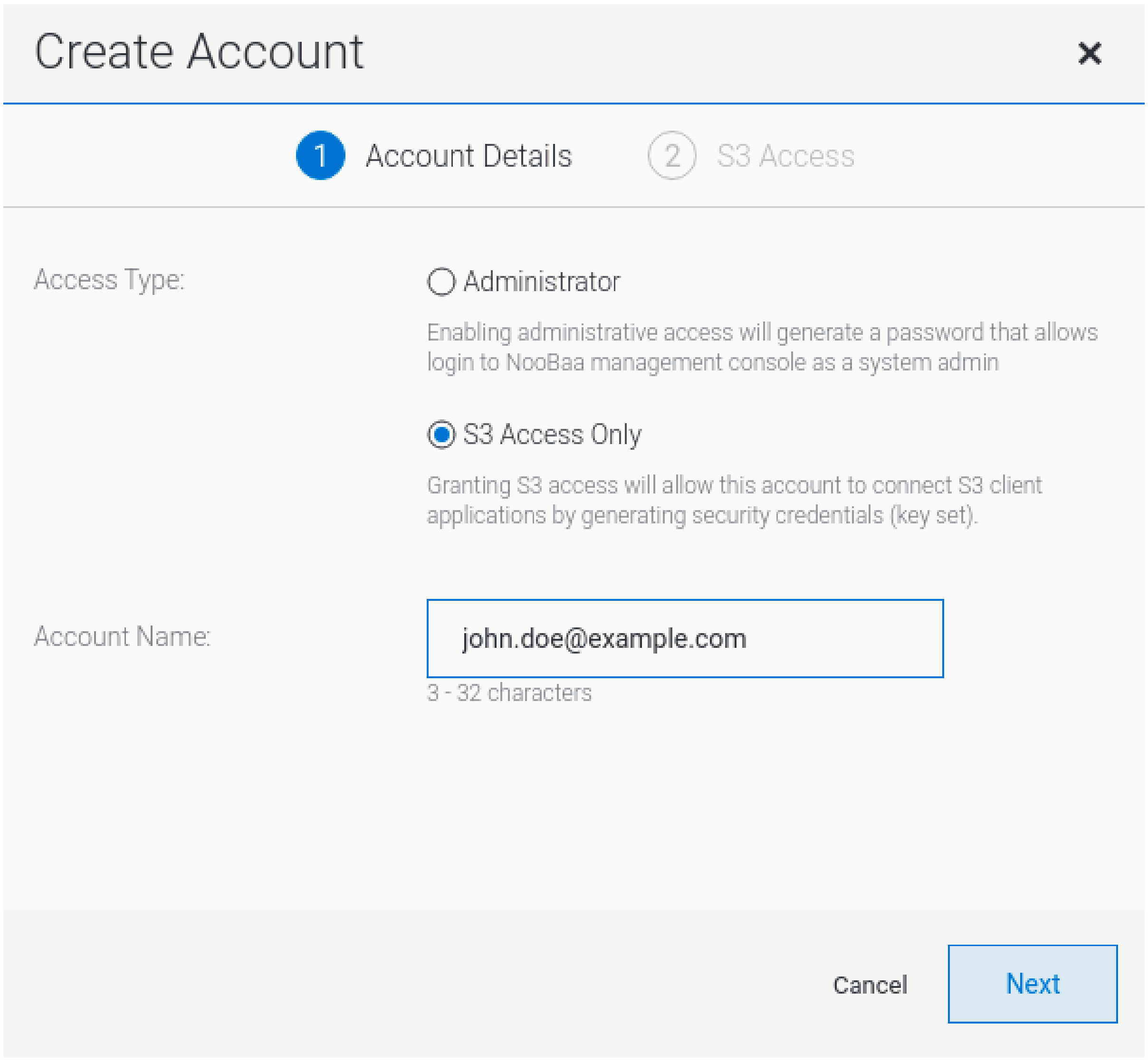
Data Consumption
Providers
I/O Operations
I/O Operations count

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



Account Name	Access Type	Default Resource
admin@noobaa.io	Administator	noobaa-default-backing-store
kube:admin (Current user)	Administator	noobaa-default-backing-store

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


Create Account

1 Account Details 2 S3 Access

Access Type: Administrator [✓ S3 Access Only]

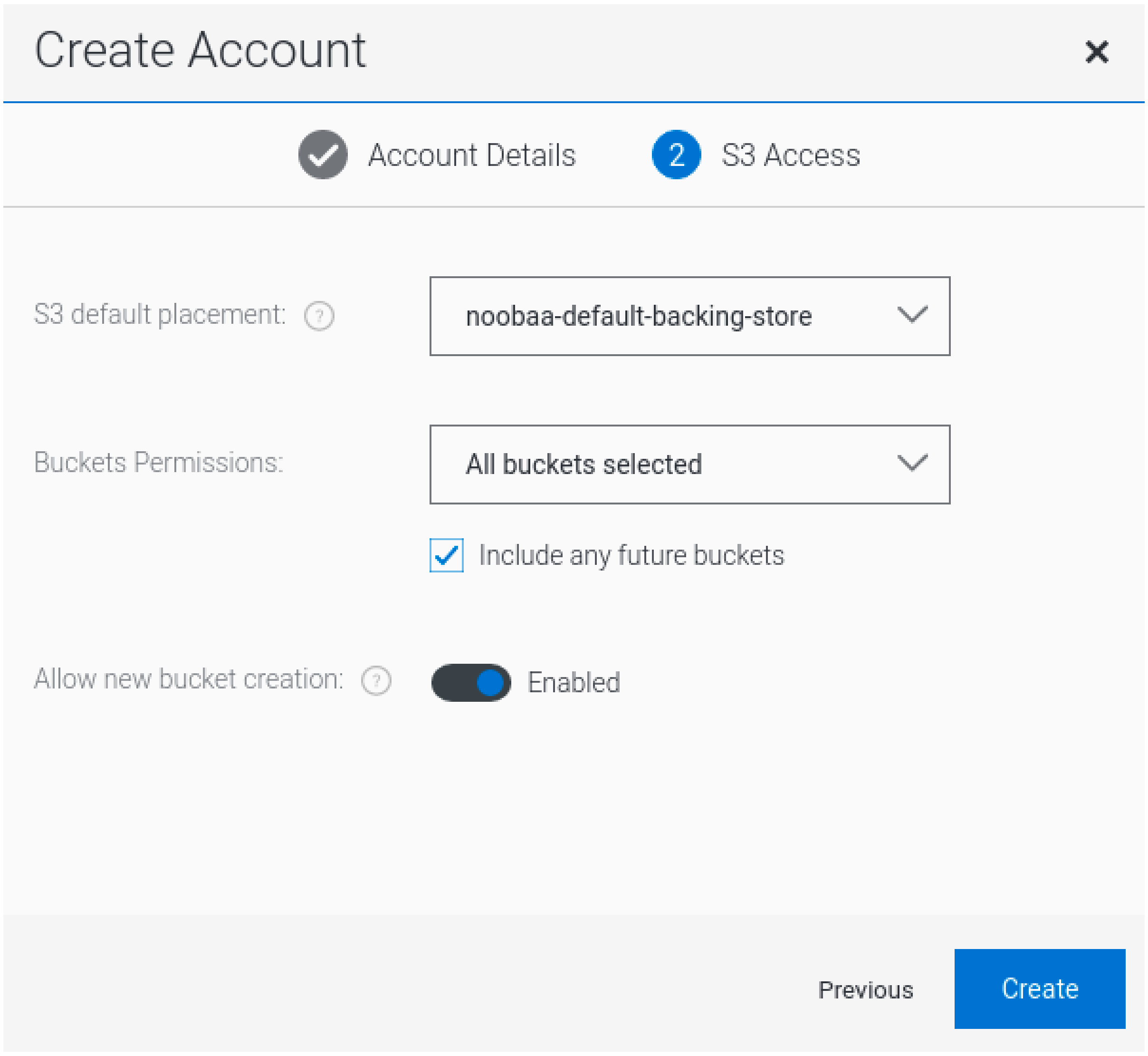
Enabling administrative access will generate a password that allows login to NooBaa management console as a system admin

Granting S3 access will allow this account to connect S3 client applications by generating security credentials (key set).

Account Name: john.doe@example.com

3 - 32 characters

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**:


Create Account

Account Details

2 S3 Access

S3 default placement: noobaa-default-backing-store

Buckets Permissions: All buckets selected

✓ Include any future buckets

Allow new bucket creation: Enabled

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

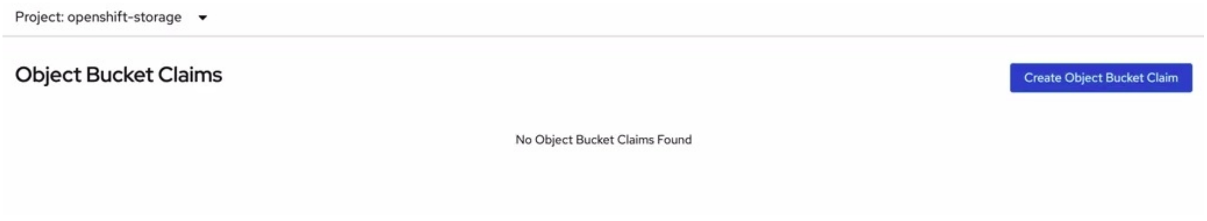
You can create an Object Bucket Claim (OBC) using the OpenShift Web Console.

Prerequisites

- Administrative access to the OpenShift Web Console.
- 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" .

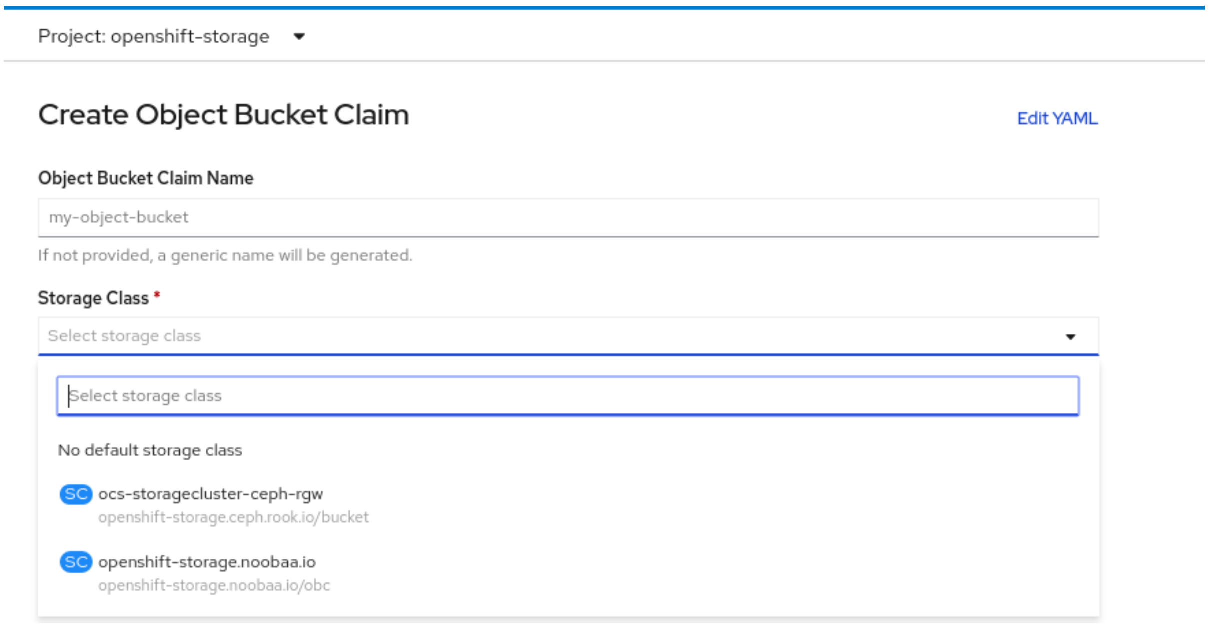
Procedure

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



No Object Bucket Claims Found

4. Enter a name for your object bucket claim and select the appropriate storage class from the dropdown menu:



Project: openshift-storage

Create Object Bucket Claim

Edit YAML

Object Bucket Claim Name

my-object-bucket

If not provided, a generic name will be generated.

Storage Class *

Select storage class

Select storage class

No default storage class

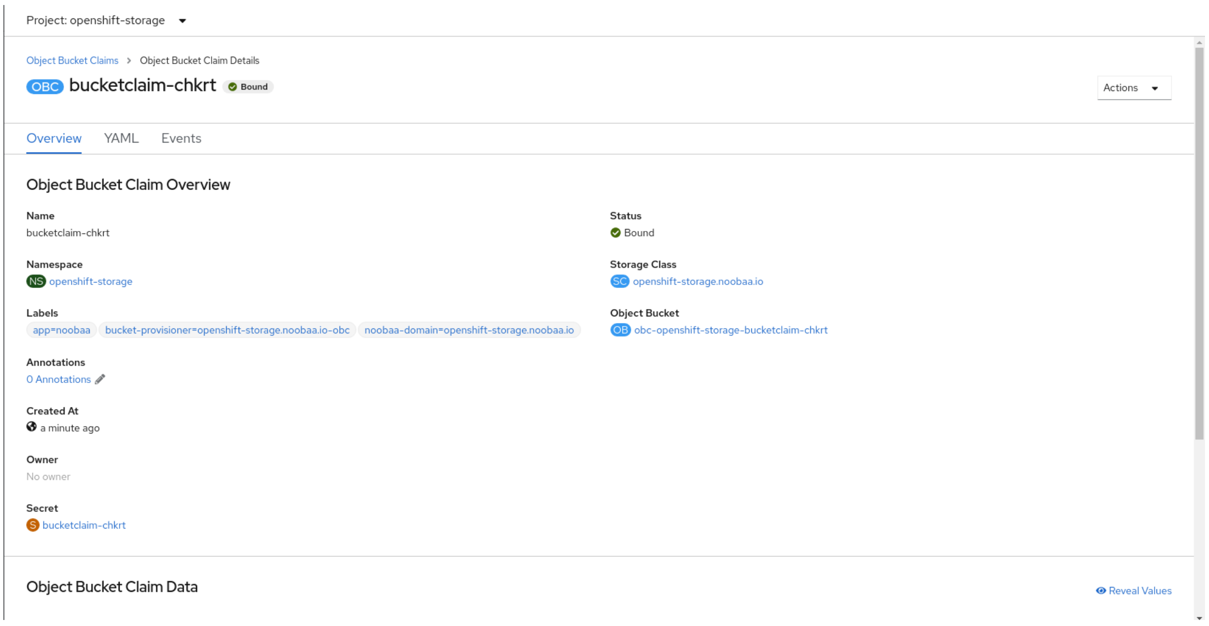
SC ocs-storagecluster-ceph-rgw
openshift-storage.ceph.rook.io/bucket

SC openshift-storage.noobaa.io
openshift-storage.noobaa.io/obc

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

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

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



Project: openshift-storage

Object Bucket Claims > Object Bucket Claim Details

OBC bucketclaim-chkrt Bound

Actions

Overview YAML Events

Object Bucket Claim Overview

Name
bucketclaim-chkrt

Status
Bound

Namespace
NS openshift-storage

Storage Class
SC openshift-storage.noobaa.io

Labels
app=noobaa bucket-provisioner=openshift-storage.noobaa.io-obc noobaa-domain=openshift-storage.noobaa.io

Object Bucket
OB obc-openshift-storage-bucketclaim-chkrt

Annotations
0 Annotations

Created At
a minute ago

Owner
No owner

Secret
S bucketclaim-chkrt

Object Bucket Claim Data

Reveal Values

Additional Resources

- Section 7.6, "Object Bucket Claim"

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

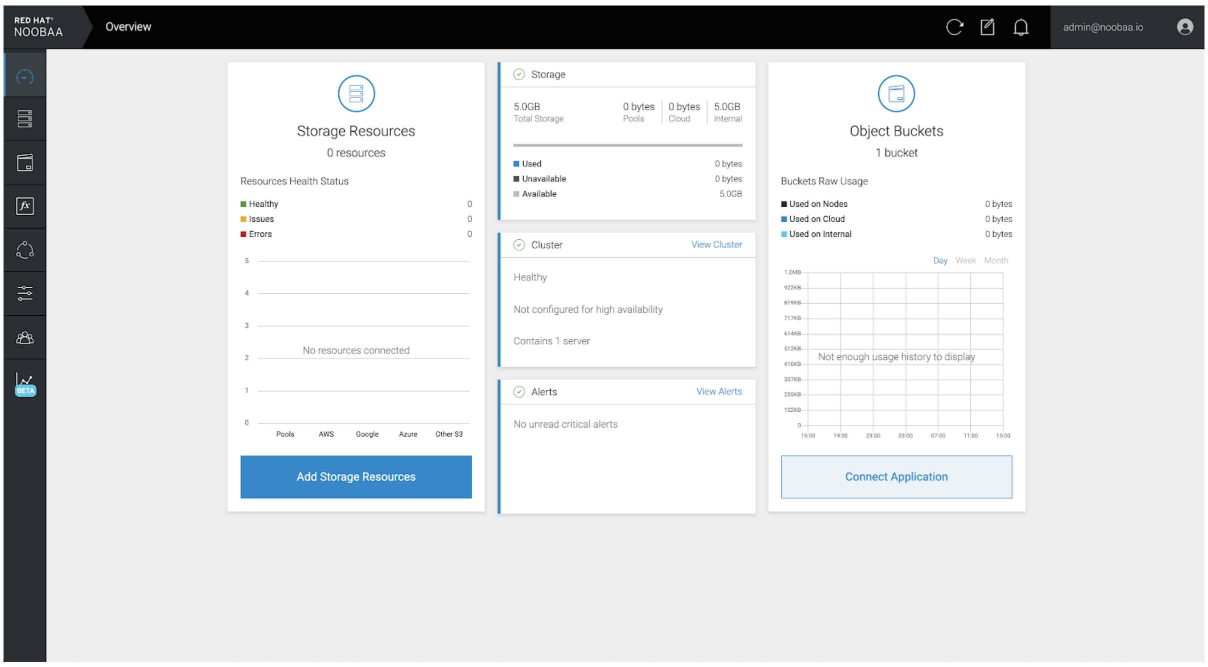
Prerequisites

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

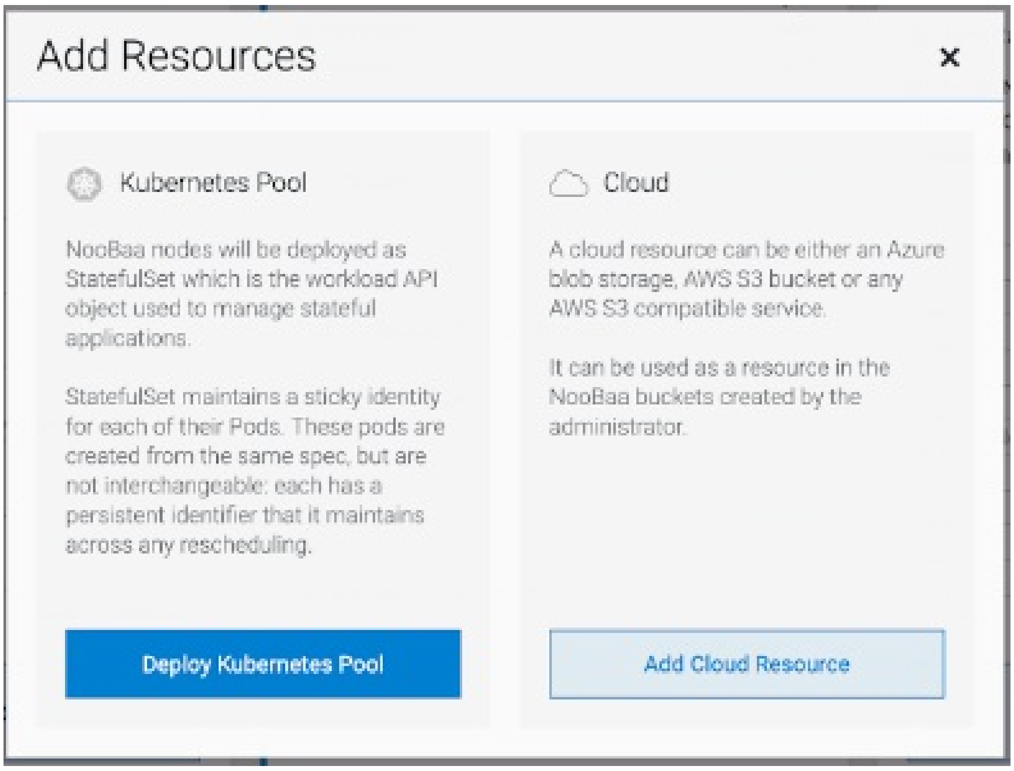
A storage node in the Multicloud Object Gateway is a NooBaa daemon container attached to one or more Persistent Volumes and used for local object service data storage. NooBaa daemons can be deployed on Kubernetes nodes. This can be done by creating a Kubernetes pool consisting of StatefulSet pods.

Procedure

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



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


Add Resources

Kubernetes Pool

NooBaa nodes will be deployed as StatefulSet which is the workload API object used to manage stateful applications.

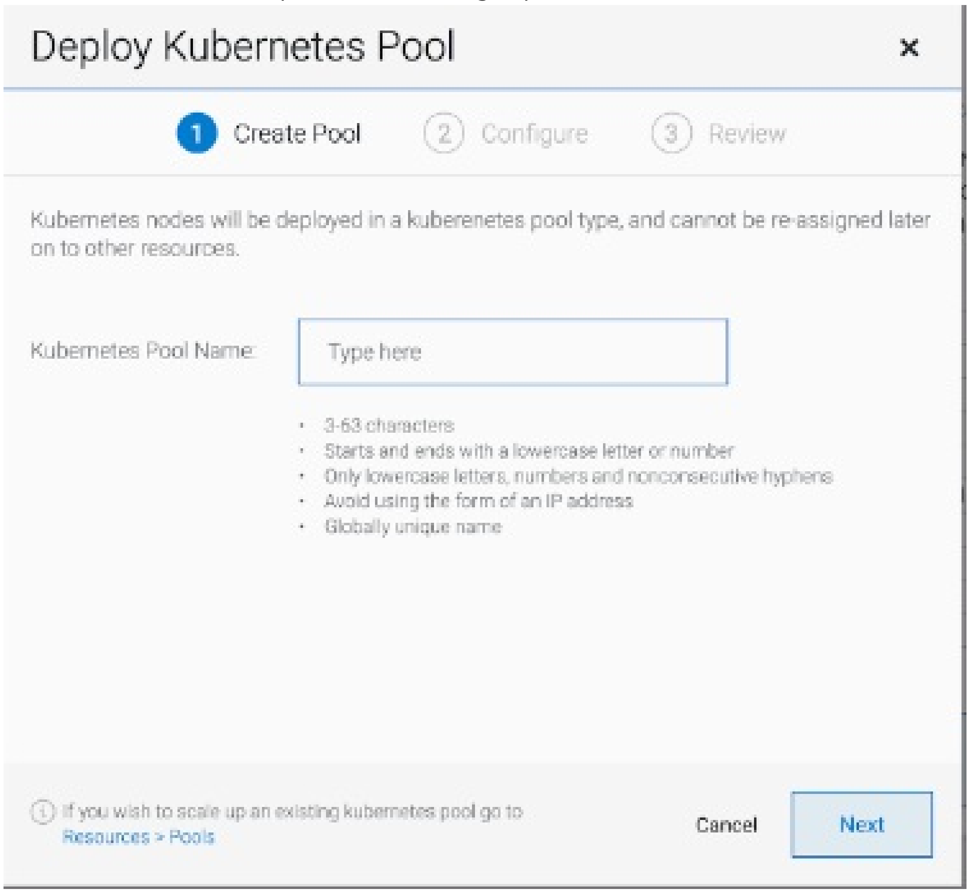
StatefulSet maintains a sticky identity for each of their Pods. These pods are created from the same spec, but are not interchangeable: each has a persistent identifier that it maintains across any rescheduling.

Cloud

A cloud resource can be either an Azure blob storage, AWS S3 bucket or any AWS S3 compatible service.

It can be used as a resource in the NooBaa buckets created by the administrator.

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


Deploy Kubernetes Pool

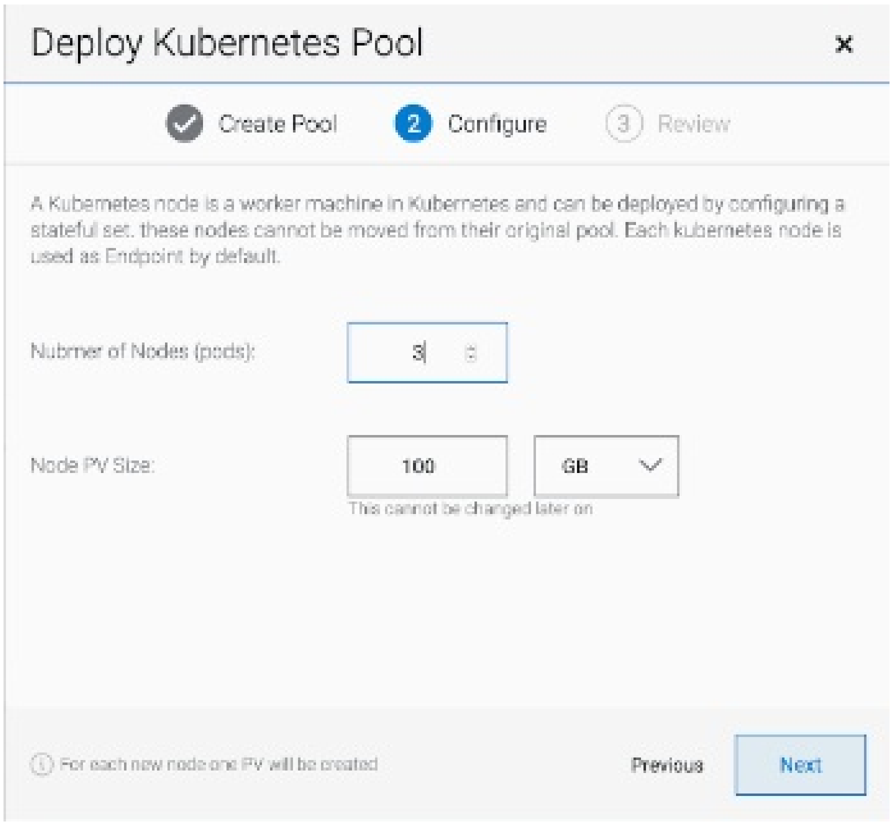
Kubernetes nodes will be deployed in a kuberenetes pool type, and cannot be re-assigned later on to other resources.

Kubernetes Pool Name: Type here

- 3-63 characters
- Starts and ends with a lowercase letter or number
- Only lowercase letters, numbers and nonconsecutive hyphens
- Avoid using the form of an IP address
- Globally unique name

If you wish to scale up an existing kubernetes pool go to Resources > Pools

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.



A Kubernetes node is a worker machine in Kubernetes and can be deployed by configuring a stateful set. these nodes cannot be moved from their original pool. Each kubernetes node is used as Endpoint by default.

Nubmer of Nodes (pods): 3

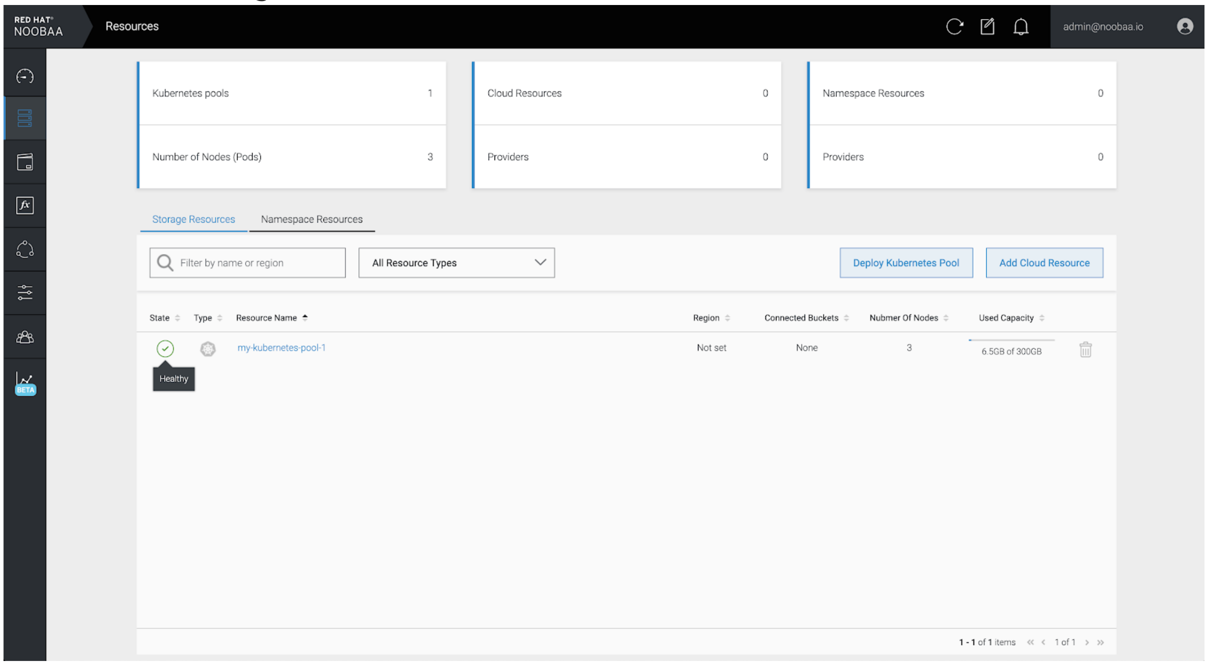
Node PV Size: 100 GB

This cannot be changed later on

For each new node one PV will 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**:



Kubernetes pools	1
Number of Nodes (Pods)	3
Cloud Resources	0
Providers	0
Namespace Resources	0
Providers	0

State	Type	Resource Name	Region	Connected Buckets	Nubmer Of Nodes	Used Capacity
✓		my-kubernetes-pool-1	Not set	None	3	6.5GB of 300GB

1 - 1 of 1 items 1 of 1

CHAPTER 8. MANAGING PERSISTENT VOLUME CLAIMS

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

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

Prerequisites

- Administrator access to OpenShift Container Storage.

Procedure

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

3. Search for the required PVC name by using the **Filter** textbox. You can also filter the list of PVCs by Name or Label to narrow down the list
4. Check the **Status** column corresponding to the required PVC.
5. Click the required **Name** to view the PVC details.

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. EXPANDING PERSISTENT VOLUME CLAIMS

OpenShift Container Storage supports expansion of Persistent Volume Claims to provide flexibility to the management of persistent storage resources.

Expansion is supported for the following Persistent Volumes:

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

IMPORTANT

Expanding Persistent Volumes 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.

For more informaton, see Technology Preview Features Support Scope .

WARNING

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

NOTE

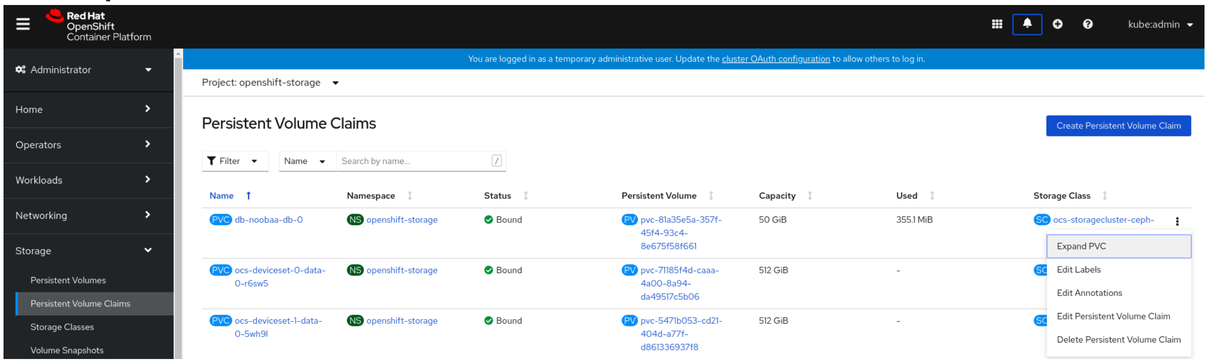
This Technology Preview feature is only available with fresh installations of OpenShift Container Storage version 4.5. It does not apply to clusters upgraded from previous OpenShift Container Storage releases.

Prerequisites

- Administrator access to OpenShift Web Console.

Procedure

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



Red Hat OpenShift Container Platform

kube:admin

Administrator

Home

Operators

Workloads

Networking

Storage

Persistent Volumes

Persistent Volume Claims

Storage Classes

Volume Snapshots

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: openshift-storage

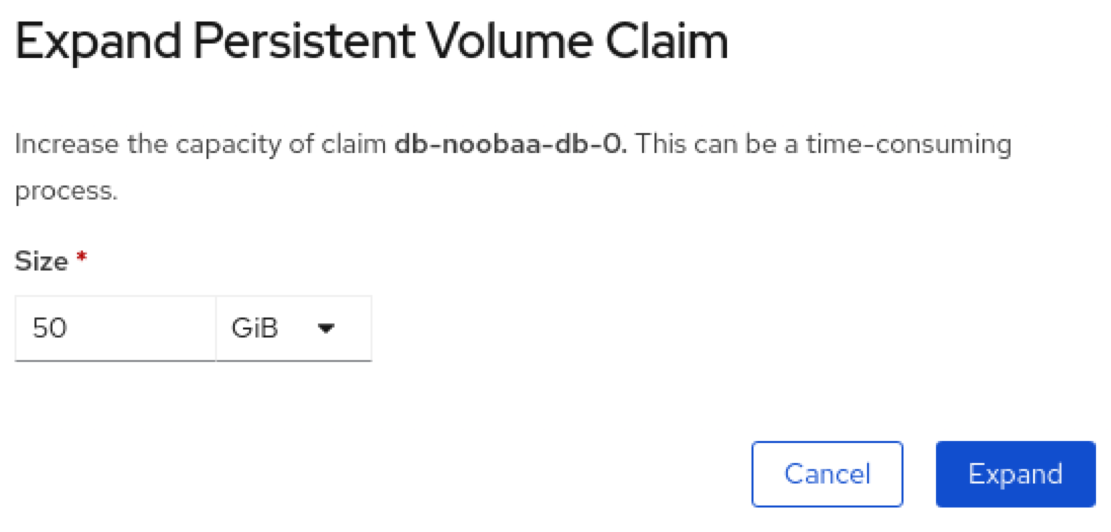
Persistent Volume Claims

Create Persistent Volume Claim

Filter | Name | Search by name...

Name	Namespace	Status	Persistent Volume	Capacity	Used	Storage Class
PVC db-noobaa-db-0	NS openshift-storage	Bound	PV pvc-81a35e5a-357f-45f4-93c4-8e675f58f661	50 GiB	355.1 MiB	SC ocs-storagecluster-ceph-
PVC ocs-deviceset-0-data-0-r6sw5	NS openshift-storage	Bound	PV pvc-71185f4d-caaa-4a00-8a94-da49517c5b06	512 GiB	-	SC
PVC ocs-deviceset-1-data-0-5wh9l	NS openshift-storage	Bound	PV pvc-5471b053-cd21-404d-a77f-d861336937f8	512 GiB	-	SC

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


Expand Persistent Volume Claim

Increase the capacity of claim **db-noobaa-db-0.** This can be a time-consuming process.

Size *

50 GiB

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

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

8.5. DYNAMIC PROVISIONING

8.5.1. About dynamic provisioning

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

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

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

8.5.2. Dynamic provisioning in OpenShift Container Storage

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

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

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

Version 4.5 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 4.5, 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.

8.5.3. Available dynamic provisioning plug-ins

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

Storage type	Provisioner plug-in name	Notes
OpenStack Cinder	**kubernetes.io/cinder**	
AWS Elastic Block Store (EBS)	**kubernetes.io/aws-ebs**	For dynamic provisioning when using multiple clusters in different zones, tag each node with **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>** where **<cluster_name>** and **<cluster_id>** are unique per cluster.
AWS Elastic File System (EFS)		Dynamic provisioning is accomplished through the EFS provisioner pod and not through a provisioner plug-in.

Storage type	Provisioner plug-in name	Notes
Azure Disk	**kubernetes.io/azure-disk**	
Azure File	**kubernetes.io/azure-file**	The **persistent-volume-binder** ServiceAccount requires permissions to create and get Secrets to store the Azure storage account and keys.
GCE Persistent Disk (gcePD)	**kubernetes.io/gce-pd**	In multi-zone configurations, it is advisable to run one OpenShift Container Platform cluster per GCE project to avoid PVs from being created in zones where no node in the current cluster exists.
VMware vSphere	**kubernetes.io/vsphere-volume**	

IMPORTANT

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

CHAPTER 9. REPLACING FAILED STORAGE NODES ON RED HAT VIRTUALIZATION PLATFORM

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

Prerequisites

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

Procedure

1. Identify the node and get labels on the node to be replaced.

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

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

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

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

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

4. Mark the nodes as unschedulable.

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

5. Drain the node.

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

NOTE

If the failed node is not connected to the network, remove the pods running on it by using the command:

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

6. Remove the failed node after making a note of the device **by-id**.

   a. List the nodes.

```
$ oc get nodes
```

Example output:

```
NAME         STATUS   ROLES    AGE     VERSION
rhvworker01    Ready    worker   6h45m   v1.16.2
rhvworker02    Ready    worker   6h45m   v1.16.2
rhvworker03    Ready    worker   6h45m   v1.16.2
```

b. Find the unique **by-id** device name for each of the existing nodes.

```
$ oc debug node/<Nodename>
```

Example output:

```
NAME                        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                         8:0    0   120G  0 disk
|-sda1                       8:1    0   384M  0 part /boot
|-sda2                       8:2    0   127M  0 part /boot/efi
|-sda3                       8:3    0     1M  0 part
`-sda4                       8:4    0 119.5G  0 part
  -coreos-luks-root-nocrypt 253:0    0 119.5G  0 dm   /sysroot
sdb                         8:16   0   500G  0 disk
sr0                        11:0    1   374K  0 rom
sr1                        11:1    1  1024M  0 rom
rbd0                       252:0    0    30G  0 disk /var/lib/kubelet/pods/1fe52fb8-ca70-40e1-
ac74-28802baf3b80/volumes/kubernetes.io~csi/pvc-705982e5-0a7d-42f5-916f-
69340541d52d/mount
rbd1                       252:16   0    30G  0 disk /var/lib/kubelet/pods/b48b22ba-532f-4d4f-
9f58-debfd03c3221/volumes/kubernetes.io~csi/pvc-bb7cdb1c-03fe-4b31-97e3-
3c8fa9758f16/mount
sh-4.4# ls -l /dev/disk/by-id/ | grep sdb
lrwxrwxrwx. 1 root root  9 Oct  7 14:58 scsi-0QEMU_QEMU_HARDDISK_a01cab90-
d3cd-4822-a7eb-a5553d4b619a -> ../../sdb
lrwxrwxrwx. 1 root root  9 Oct  7 14:58 scsi-SQEMU_QEMU_HARDDISK_a01cab90-
d3cd-4822-a7eb-a5553d4b619a -> ../../sdb
sh-4.4# exit
exit
sh-4.2# exit
exit
Removing debug pod ...
```

Repeat this to identify the device ID for all the existing nodes.

c. Delete the machine corresponding to the failed node. A new node is automatically added.

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

   ii. Beside the required machine, click the Action menu **(⋮) → Delete Machine**

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

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

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

d. [Optional]: If the failed Red Hat Virtualization instance is not removed automatically, terminate the instance from the console.

7. Click **Compute** → **Nodes** in the OpenShift web console. 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. Add the local storage devices available in the new worker node to the OpenShift Container Storage StorageCluster.

   a. Add the new disk entries to LocalVolume CR.
   Edit **LocalVolume** CR. You can either remove or comment out the failed device **/dev/disk/by-id/{id}** and add the new **/dev/disk/by-id/{id}**. For information about identifying the device **by-id**, see Finding available storage devices.

   ```
   $ oc get -n local-storage localvolume
   ```

   Example output:

   ```
   NAME          AGE
   local-block   25h
   ```

   ```
   $ oc edit -n local-storage localvolume local-block
   ```

   Example output:

   ```
   [...]
     storageClassDevices:
     - devicePaths:
       - /dev/disk/by-id/scsi-SQEMU_QEMU_HARDDISK_901abab1-b164-4bb4-b9a8-
   6bdbce2de23b
       - /dev/disk/by-id/scsi-SQEMU_QEMU_HARDDISK_a01cab90-d3cd-4822-a7eb-
   a5553d4b619a
       # SQEMU_QEMU_HARDDISK_a01cab90-d3cd-4822-a7be-a5553d4b619a
       - /dev/disk/by-id/scsi-SQEMU_QEMU_HARDDISK_c7388957-3e09-4135-bac1-
   af7551467d0b
   ```

```
      storageClassName: localblock
      volumeMode: Block
[...]
```

Make sure to save the changes after editing the CR.

You can see that in this CR the following new device using **by-id** has been added.

SQEMU_QEMU_HARDDISK_a01cab90-d3cd-4822-a7eb-a5553d4b619a

b. Display PVs with **localblock**.

```
$ oc get pv | grep localblock
```

Example output:

```
local-pv-3646185e   2328Gi  RWO     Delete      Available
localblock  9s
local-pv-3933e86    2328Gi  RWO     Delete      Bound       openshift-storage/ocs-
deviceset-2-1-v9jp4   localblock  5h1m
local-pv-8176b2bf   2328Gi  RWO     Delete      Bound       openshift-storage/ocs-
deviceset-0-0-nvs68   localblock  5h1m
local-pv-ab7cabb3   2328Gi  RWO     Delete      Available
localblock  9s
local-pv-ac52e8a    2328Gi  RWO     Delete      Bound       openshift-storage/ocs-
deviceset-1-0-knrgr   localblock  5h1m
local-pv-b7e6fd37   2328Gi  RWO     Delete      Bound       openshift-storage/ocs-
deviceset-2-0-rdm7m   localblock  5h1m
local-pv-cb454338   2328Gi  RWO     Delete      Bound       openshift-storage/ocs-
deviceset-0-1-h9hfm   localblock  5h1m
local-pv-da5e3175   2328Gi  RWO     Delete      Bound       openshift-storage/ocs-
deviceset-1-1-g97lq   localblock  5h
...
```

10. Delete each PV and OSD associated with the failed node using the following steps.

a. Identify the DeviceSet associated with the OSD to be replaced.

```
$ osd_id_to_remove=0
$ oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

where, **osd_id_to_remove** is the integer in the pod name immediately after the **rook-ceph-osd** prefix. In this example, the deployment name is **rook-ceph-osd-0**.

Example output:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
    ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

b. Identify the PV associated with the PVC.

```
$ oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

where, x, y, and pvc-suffix are the values in the DeviceSet identified in an earlier step.

Example output:

```
NAME                STATUS     VOLUME      CAPACITY   ACCESS MODES
STORAGECLASS   AGE
ocs-deviceset-0-0-nvs68   Bound   local-pv-8176b2bf   2328Gi     RWO          localblock
4h49m
```

In this example, the associated PV is **local-pv-8176b2bf**.

c. Delete the PVC which was identified in earlier steps. In this example, the PVC name is ocs-deviceset-0-0-nvs68.

```
$ oc delete pvc ocs-deviceset-0-0-nvs68 -n openshift-storage
```

Example output:

```
persistentvolumeclaim "ocs-deviceset-0-0-nvs68" deleted
```

d. Delete the PV which was identified in earlier steps. In this example, the PV name is local-pv-8176b2bf.

```
$ oc delete pv local-pv-8176b2bf
```

Example output:

```
persistentvolume "local-pv-8176b2bf" deleted
```

e. Remove the failed OSD from the cluster.

```
$ oc process -n openshift-storage ocs-osd-removal -p
FAILED_OSD_ID=${osd_id_to_remove} | oc create -f -
```

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

```
# oc get pod -l job-name=ocs-osd-removal-${osd_id_to_remove} -n openshift-storage
```

NOTE

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

```
# oc logs -l job-name=ocs-osd-removal-${osd_id_to_remove} -n openshift-
storage --tail=-1
```

g. Delete the OSD pod deployment.

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
```

11. Delete **crashcollector** pod deployment identified in an earlier step.

    ```
    $ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=
    <old_node_name> -n openshift-storage
    ```

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

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

    Example output:

    ```
    NAME                                READY   STATUS    RESTARTS   AGE
    rook-ceph-operator-6f74fb5bff-2d982   1/1     Running   0          5h3m
    ```

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

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

       Example output:

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

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

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

       Example output:

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

       Creation of the new OSD may take several minutes after the operator starts.

13. Delete the **ocs-osd-removal** job(s).

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

    Example output:

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

Verification steps

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

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

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

   - **csi-cephfsplugin-***

- **csi-rbdplugin-***

3. Verify that all other required OpenShift Container Storage pods are in **Running** state.
   Also, ensure that the new incremental **mon** is created and is in the **Running** state.

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

   Example output:

   ```
   rook-ceph-mon-a-64556f7659-c2ngc    1/1     Running     0   5h1m
   rook-ceph-mon-b-7c8b74dc4d-tt6hd    1/1     Running     0   5h1m
   rook-ceph-mon-d-57fb8c657-wg5f2     1/1     Running     0   27m
   ```

   OSDs and mons might take several minutes to get to the **Running** state.

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

CHAPTER 10. REPLACING FAILED STORAGE DEVICES ON RED HAT VIRTUALIZATION PLATFORM

When you need to replace a storage device on Red Hat Virtualization platform, you must replace the storage node. For information about how to replace nodes, see Replacing failed storage nodes on Red Hat Virtualization platform.